POLITECNICO DI TORINO

Collegio di Ingegneria Gestionale

Master final thesis

MACHINE LEARNING IN SMART FACTORIES AND FAILURE ANALYSIS

Advisors

Advisor's sign
Prof. Franco Lombardi

Advisor's sign
Dott.ssa Giulia Bruno

Candidate

Candidate's sign

Sergi Fajas Garriga

2nd July, 2018

Index

1. Introduction

This thesis is related to the concept of “new world and new opportunities” that have been appearing due to the increasing power of virtual world. Since the improvement of computer technology, all type of data can be collected and moreover, it can be treated in many ways. Getting data, transforming them and solving problems are the main objectives with the irruption of this new world. If all of this could be done, some problems of the companies would be solved in a simple way and they could take advantage of it.

On the other hand, solving problems is a tricky question, because each problem is different and it could accept more than one solution. Finding the best solution in a minimum lack of time is the key point to avoid losses in your company and it could be done with subsets of artificial intelligence. The most useful artificial intelligence in this context is machine learning. It allows to learn and improve the model that is built from an initial dataset and it could also recognize or predict some values thanks to training methods.

Some years ago, predicting values and obtaining results was not as easy as nowadays. Since the barrage of machine learning, some business are trying to adapt its infrastructures to this new virtual concept in order to achieve better results than before and consequently, increasing their efficiency. This kind of business are called Smart factories because they combine physic world with robotics world. The main aim is to join forces and to get the best result they can reach.

Machine learning, in terms of artificial intelligence, is based on getting data from some kinds of processes, then select the best algorithms to work with these data, train one attribute that will be the bottleneck of the process, and finally analyse the predictable results obtained. If the predictable results are similar or equal to the real values, then it means that all the models built are working well. Otherwise, if the results are not similar to the real values, it means that something is going wrong and the models need more training to recognize better the data and the attributes given.

Furthermore, machine learning has two types of learning methods. The first one is supervised learning and the other one is unsupervised. While the first one needs some inputs to be able to produce an output, the second one learns by itself from the moment it receives data. Figure 1.1 shows the evolution of data minig and machine learning mentioned.

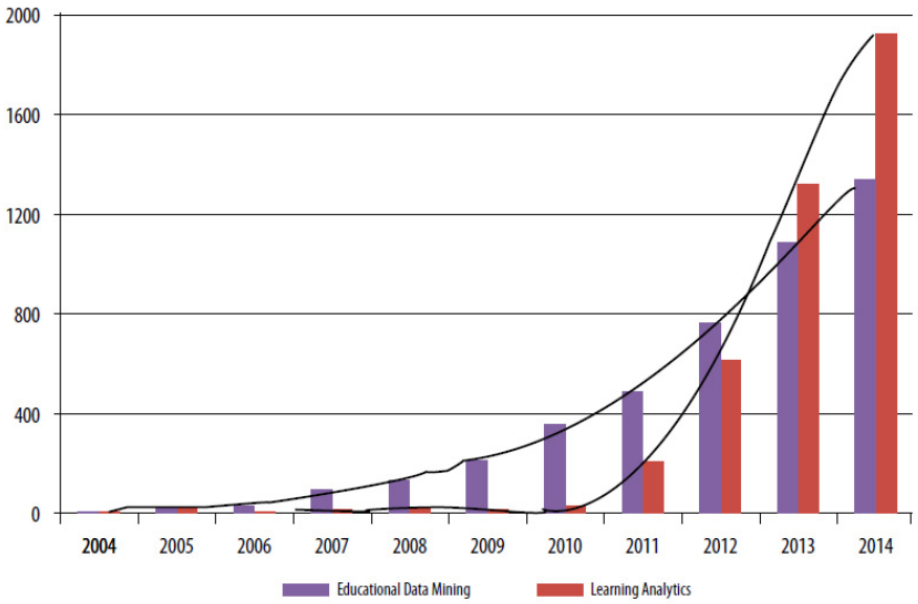


Figure 1.1: Evolution of data mining and learning analytics

1.1. Thesis objective

The objective of this thesis is to acquire more knowledge about the virtual world and how the technologies could be applied into a company to improve its gains. Solving the problems that appear daily is crucial for the future of the factories. Moreover, in Smart Factories, the key point is not to solve the problems but rather to anticipate them.

In this thesis, the main goal is to create a model to solve problems that can appear in a lot of companies. All the companies that manufactures metallic products need a tool to pull the chip out. The problem is that the tool wear could affect the production of the pieces, to the point that the costumer does not want the piece because it has some errors.

In this case, a dataset is given and, with it and Microsoft Azure a model has to be developed. The first step is to analyse all the algorithms available and choosing the best one, and then, the model should be trained to get the optimal results. Optimal results means that the predictable values are similar to the real ones. If it is achieved, then the company could understand the source of the problems and thus, predict probable mistakes from the results extracted.

2. Data mining

Over the past few decades, the increasing power of computer technology has led to a significant improvement in data collection, storage, and manipulation ability. Data mining, a technique that involves machine learning and statistical methods, is the process of finding hidden patterns [1] in large datasets. The main goal of data mining is to extract information from an input dataset and subsequently transform it into an understandable and usable structure. In other words, data mining is the automatic analysis of large quantities of data to extract previously unknown patterns [2]. Some examples of these data patterns are groups of data records (cluster analysis, section 5), dependencies detection (association analysis, section 4) and anomaly detection.

Data mining is the link between applied statistics and artificial intelligence to database management. Data is treated, stored and labelled in databases to achieve higher efficiency.

Detected patterns can be used in further analysis like machine learning and predictive analytics. A real-world example application can be seen in decision support systems[1]. The data mining step identifies multiple groups in the data, which are subsequently used to perform accurate prediction results. Data management, feature engineering, data pre-processing and post-processing are some of the steps that take place in data mining.

Data mining involves five major classes of tasks [3] . Anomaly detection or outlier detection is the identification of unusual data records. In association rule learning, relationships between variables are detected. Clustering is the task of discovering similar groups and structures in the data, without using previously known structures in data. Classification is the task of performing a generalization to some known structure and applying to new data afterward. In regression, a function modelling the data with the minimum error is found. This function estimates the relationships between independent or fix variables and the dependent variable.

[1] A decision support system (DSS) is an information system that supports business or organizational decision-making activities. DSSs serve the management, operations and planning levels of an organization.

3. Classification

Classification is the process of assigning labels to previously unlabelled intance by means of a model (classifier). The model has the so-called training set, which is a set of intance containing attribute values and class labels each one [4].

A classification model can be used in two different ways. Either as a predictive model to classify previously unlabelled intance, or as a descriptive model to identify the characteristics that distinguish intance from different classes.

The process of learning a classification model from a given training set is called learning algorithm. The process of using a learning algorithm to build a classification model from the training data is known as induction. In contrast, the process of applying a classification model to unlabelled intance to predict their class labels is known as a deduction.

It is important to properly separate the training set from the test set, in order to ensure that the model can correctly predict the class label of previously unseen intance. By a table called confusion matrix, the performance of a model can be evaluated. This table consists of a comparison between predicted labels and true labels of intance. Likewise, evaluation metrics such as accuracy or error rate can be extracted from the information in the confusion matrix.

3.1. Artificial Neural Network

NN or Artificial Neural Networks are inspired by the functionality of the brain. More precisely, NN simulate the decentralized 'computation' of the central nervous system by parallel processing and allowing an artificial system to perform some machine learning task, e.g. pattern recognition. Nowadays, NN are applied to several fields of manufacturing, for example in the semiconductor manufacturing and, in general, in process control. NN are capable of handling high-dimensional and multivariate data, however, sometimes they can suffer from over-fitting and incorporate high-variance algorithms.

Basically, this algorithm consists of input and output layers and, in most cases, there are also hidden layers whose function is to transform input data in something that the output layer could use. Moreover, it is very useful to detect some patterns that could be hidden, and this algorithm analyses quite well the most complex systems.

Even though NN were first studied 70 years ago, it is now that they are taking some value in machine learning world due to the *backpropagation,* which is allowing NN to adjust their hidden layers when an unexpected outcome is obtained. These hidden layers have an important role since they must change the data of the output. Sometimes the output of each artificial neuron is computed by some non-linear function of the sum of its inputs [5].

In addition, with the advent of deep learning neural network, it is possible to mix some multilayers formed by more than one layer, where each of them can extract some features until the algorithm learns. Likewise, it is possible to know which pieces of evidence are.

NN models can be viewed as mathematical models defining a function $f: \mathrm{X} \rightarrow \mathrm{Y}$. Normally, a neuron network's function $f(x)$ is defined as a composition of other functions $g_i(x)$ and at the same time, they are decomposed in some others. As there are some functions that are related, the algorithm has to be represented as a network structure with arrows to make clear all the relations that involve. In Figure 3.1 this structure is shown [6].

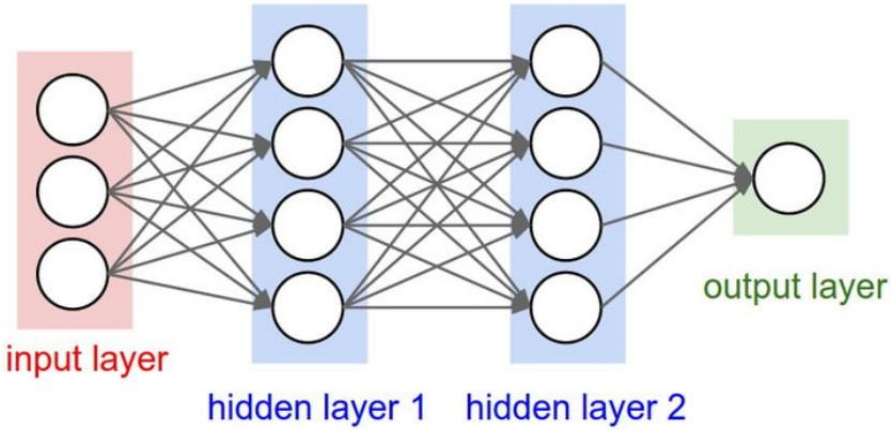


Figure 3.1: Neural Network structure

3.1.1. Types of Neural Networks

There are multiple types of neural networks and each of them have their complexity's level. Some of the most important types are exposed [7].

- **Deep feed forward (DFF):** is one of the most used neural networks because its information travels in only one direction, from the input to output. Furthermore, all the nodes are fully connected, and it allows to extract a great deal of information. In addition, the activation flow goes from the input layer to the output, without any black loop that could worsen the final results.

- **Recurrent Neural Network (RNN):** this method is used when a decision from past iterations or samples can influence the output or the final result. Since RNN possess greater learning abilities, it is normally used when the system is more complex. It is used for language and handwriting recognition as it has recurrent cells that allow using the decisions from the past. Figure 3.2 shows it.

- **Auto Encoder (AE):** this method is basically used for classification, clustering and feature compression. It can be trained without supervision and once the training is done, it should be able to search for common patterns and generalize data. Figure 3.3 shows it.

- **Restricted Boltzmann Machine (RBM):** it is similar to Boltzmann Machine neural network because some cells are marked as an input and remain hidden, and other cells become an output, as soon as each hidden cell update their state. Normally, these changes are done when the model is being trained.

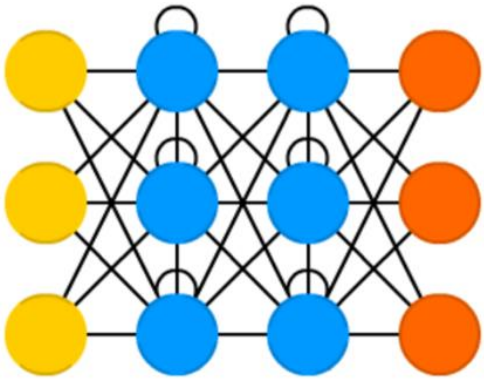


Figure 3.2: Recurrent Neural Network

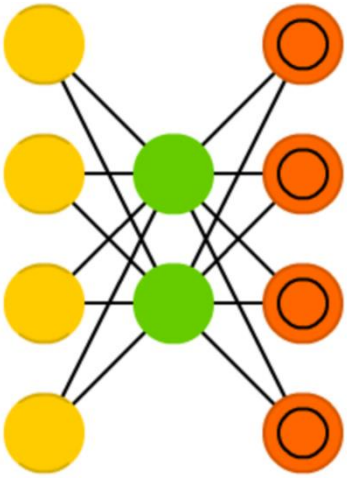


Figure 3.3: Auto Encoder

3.2. Support Vector Machines

Another rather new and promising ML algorithm with generally high performance, the ability to achieve high accuracy, the ability to handle high-dimensional multi-variate datasets, and good results in some practical manufacturing applications is *Support Vector Machines* (SVMs). Nowadays, there is a wide range of SVMs applications to solve real world problems. SVM is a good choice when it comes to text categorization, classification of images, handwriting recognition, and biological science applications, among others.

SVMs are supervised learning models with learning algorithms that analyse data used for classification and regression analysis. With this algorithm training data is separated into two

categories, representing two distinct class labels either one. First, an n-dimensional space is created depending on the length of the input vector; second, dataset belonging to either class are separated by a plane through space, and thus labelled according to the side of the plane where are located (Figure 3.4). Hence, SVMs are non-probabilistic binary linear classifiers. In other words, SVMs is a representation of datasets as points in space, mapped so they are divided by a clear gap as wide as possible. Input datasets are subsequently predicted to belong to one category depending on which side of the gap they fall in.

As stated, a SVM creates a hyperplane or set of hyperplanes in a high or even infinite-dimensional space. The largest the distance to the nearest training-data point (called functional margin), the better the separation between classes is achieved, thus the lower the generalization error of the classifier [8].

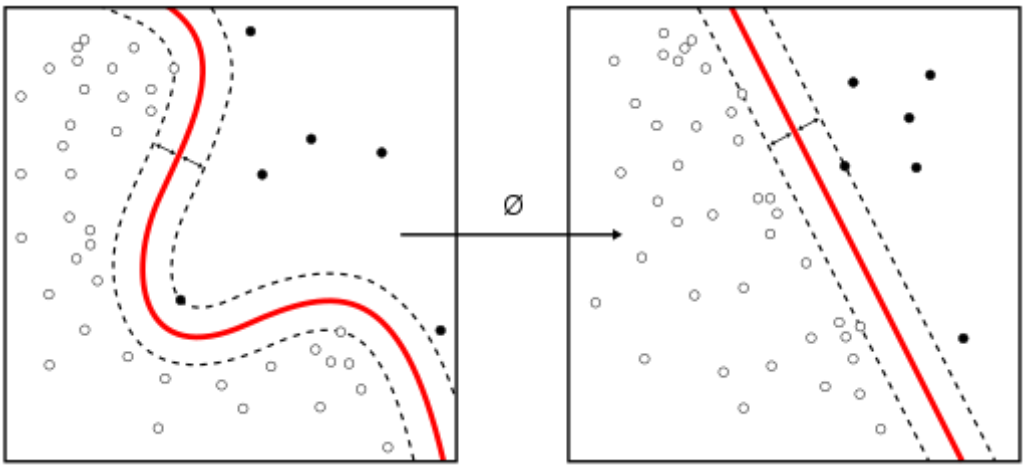


Figure 3.4: N-dimensional space

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick [9]. Thanks to kernel functions inputs can be mapped into high-dimensional feature spaces with no need of computing the coordinates of the data in that space, which would imply a high computational cost. Instead, inner products between the images of all pairs of data in the feature space are computed.

When the input datasets are unlabelled, an unsupervised learning method is required. The objective is to find natural clustering of the data in order to fit new data into one of these formed groups. The support vector clustering [10] applies support vectors to categorize unlabelled data.

3.3. Decision Tree Classifier

A well-known classification technique based on a hierarchical structure is the decision tree. The tree

has three types of nodes: a root node, with no incoming links and zero or more outgoing links (children); internal nodes, with one incoming link and two or more outgoing links (children); and leaf or terminal nodes, with one incoming link and no outgoing links. While every internal node is associated with an attribute test condition, every leaf node refers to a class label. Links between nodes refer to a possible outcome of a certain internal node.

3.3.1. Hunt's Algorithm

One recent algorithm to build a decision tree is Hunt's algorithm. Many current implementations of decision tree classifiers such as ID3, C4.5, and CART use such algorithm as a basis. In this algorithm, a decision tree is grown recursively. It starts with only one root node containing intance from different classes. As long as the node has labels of more than one class, it is expanded using an attribute test condition determined by a splitting criterion. Successively, a child leaf node is created for each possible outcome of the attribute test condition and the intance respectively distributed. The decision tree grows until all the intance associated with a leaf node have the same class label.

3.3.2. Methods for expressing attribute test conditions

Binary attributes. In this type of attributes two potential outcomes are generated by a test condition.

Nominal attributes. There are two ways of expressing attribute test conditions. First, as a multiway split with the exact number of outcomes as the number of distinct values for the attribute. Second, as a binary split by partitioning all possible values from the nominal attribute into two groups. For instance, CART algorithm produces only binary splits.

Ordinal attributes. In this type of attribute, multiway and binary splits can be created as well, provided the grouping does not change the order property of the attribute values.

Continuous attributes. For continuous attributes, the attribute test condition can be expressed as a comparison test producing a binary split, or as a range query producing a multiway split. Below an example is provided of the two mentioned splitting methods. Figure 3.5 shows it.

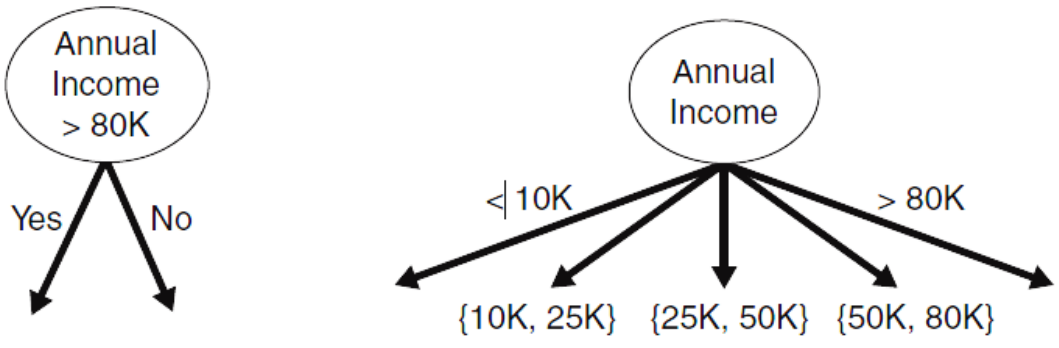


Figure 3.5: Test condition for continuous attributes.

3.4. Measures for selecting an attribute test condition

Measures for selecting an attribute test condition prioritise those attributes test condition which tends to group training intance into purer subsets (from the same class label) in the child nodes. This is particularly interesting since having child nodes with intance belonging to a unique class label implies the end of the expansion of the algorithm. On the contrary, having child nodes made up of intance from multiple classes leads to an inefficient increase in the depth of the tree.

3.4.1. Impurity Measure for a Single Node

The impurity of a node measures the dissimilarity of the class labels of the intance belonging to a certain node. Entropy, Gini index and classification error are some of the measures that can be used to determine the impurity of a certain node. All three measures give a zero-impurity value if a node contains intance from a single class and maximum impurity if the node has an equal proportion of intance from multiple classes. Figure 3.6 shows the relationship among these three measures is show.

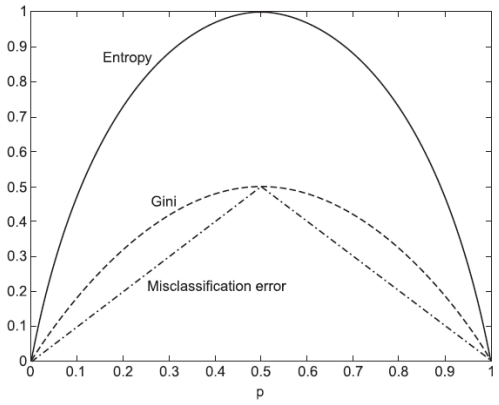


Figure 3.6: Comparison of the impurity measures

3.4.2. Collective Impurity of Child Nodes

The collective impurity of the child nodes can be calculated as the weighted sum of the impurities of the child nodes. Let's consider having N training intance, $N(v_j)$ as the number of training intance associated with a child node v_j, and the impurity value $I(v_j)$ of that child node. The collective weighted impurity of child nodes is calculated as Figure 3.7 shows.

$$I(\text{children}) = \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j),$$

Figure 3.7: Impurity of child nodes

3.4.3. Identifying the best attribute test condition

In order to identify the best attribute condition, it is necessary to compare impurities between parent node (before splitting) and child nodes (the weighted impurity after splitting). The difference in values of impurity is known as a gain in impurity of an attribute test condition, and can be expressed as follows:

$$\Delta = I(\text{parent}) - I(\text{children})$$

The higher the gain, the purer are the classes in the child nodes associated with a certain parent node, and thus the better is the test condition. The splitting criterion chooses the attribute test condition with the highest gain in impurity. Since all child nodes share the same parent node, which means the splitting criterion selects the child node with the lowest weighted impurity measure.

One drawback of some impurity measures, such as Gini index and entropy, is that they tend to prioritise qualitative attributes with a larger number of distinct values. Therefore, a low weighted child impurity value is not enough to select the most convenient attribute test condition for a node. Thus, while deciding the best attribute condition, it is imperative to consider the number of children produced by the splitting attribute. One way to overcome this problem is to use a binary algorithm such as CART. For other decision tree classifiers such as C4.5, a measure known as a gain ratio is used to compensate for parent nodes with an uneven number of child nodes.

3.5. Model overfitting

Even if a model fits well over the training data, it can still show poor generalization performance, a phenomenon known as model overfitting.

As long as the size of the tree (the number of leaf nodes) does not exceed approximately 8 nodes, both the training and the test error rate diminish as the number of leaf nodes increases. In fact, building decision tree with fewer than 3 or 4 leaf nodes leads to significant test and training errors. This phenomenon is known as model underfitting. In such simplistic trees, the model does not fully represent the relationship between the attributes and the class labels.

On the other hand, when considering decision trees with more than 8 leaf nodes, an adverse effect occurs at a certain tree size. Even though the training error decreases as the tree expands, beyond approximately 50 leaf nodes, the test error begins to increase. What is more, the gap between both errors keeps on widening as the tree size increases. This is the so-called model overfitting. To illustrate these effects, the error rate for different tree sizes is shown below in Figure 3.8 and Figure

3.9

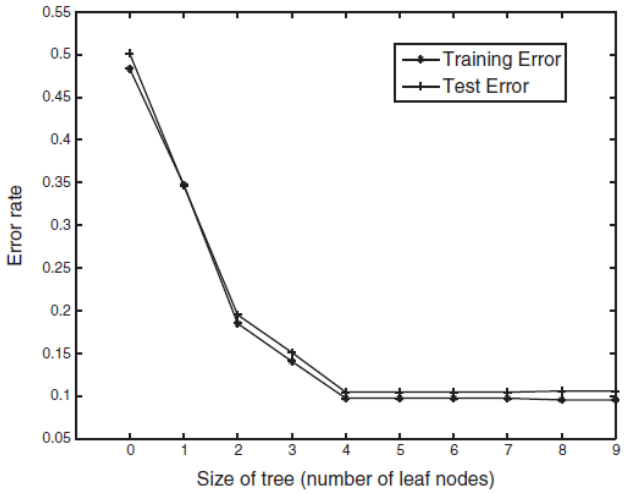


Figure 3.8: Varying tree size from 1 to 8.

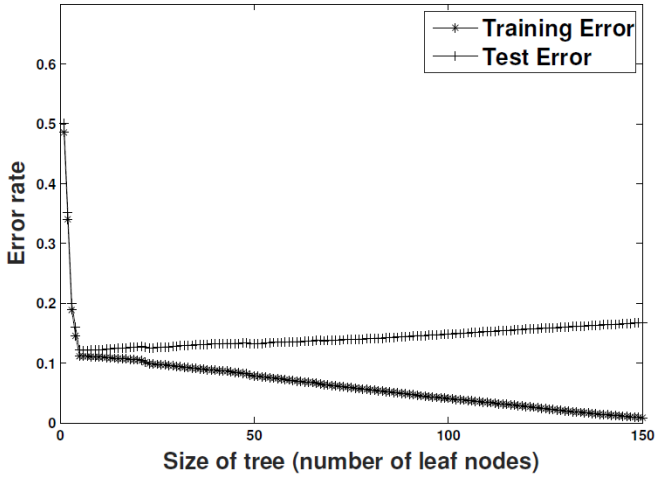


Figure 3.9: Varying tree size from 1 to 150.

3.6. Model Selection

Model selection is the process that allows you to select the right level of complexity depending on the topic you are researching. This model has to choose the best approach, otherwise the problem will not be solved successfully. Note that the model selection works with the training set.

There are three generic approaches to estimate the generalization performance of a model, and these could be the following: using a Validation Set, incorporating Model Complexity and

incorporating Model Complexity.

3.6.1. Using a Validation Set

One of the approaches to select the right model is by using the validation set. Using this, all the training sample obtained has to be divided into two groups. The first one should be a group of samples to train the model, while the other is the sample used by estimating whether the model is correct or not. Once all the data is divided into two groups, two errors rates appear. One of them will be from the sample called validation set and the other one will come from the model that is being trained. To achieve a correct approach to detect errors, those from validation set must be observed instead of those of the training's error. If training's error is used, a mistake is made because the model has been trained with this sample. Finally, the size of the samples cannot be too small because if it occurs, the model will not be accurately estimated.

3.6.2. Incorporating Model Complexity

This strategy is inspired by the principle called Occam's razor, also known the principle of parsimony. In a few words, Occam's razor claims that if there are some models explaining a problem, the simplest model will be chosen instead of choosing the others that are more complex.

- **Estimating the Complexity of Decision Trees**

  As regards the decision trees, the complexity of them is estimated as the ratio of the number of leaf nodes and the number of training intance. The training sample have a great impact on the resulting number of leaf nodes and complexity of the tree.

- **Minimum Description Length Principle**

  This strategy claims that if there is an information transfer between two subjects, the model's complexity will change depending on the accuracy of the transmission. In other words, if the transmission is not efficient, the "cost" of the model will be higher and then it will be more complex and difficult to solve.

3.6.3. Estimating Statistical Bounds

The last alternative is to apply a statistical correction to the training error rate of the model. The error indicates the model's complexity and, moreover, it only can be done if the probability distribution of the training error is known or can be assumed by anyway. Otherwise, this method is not viable.

3.6.4. Model Selection for Decision Trees

Unlike other section, where the best approaches to find a model are examied, in this section, the aim is to determine the two most common model selection strategies for decision tree induction.

- **Pre-pruning (Early Stopping Rule)**

  This strategy does not allow for exploring all the candidates that could appear. For this reason, this tree-growing algorithm is stopped before generating all the leaf nodes that perfectly fits the entire training data. The action of cutting a branch could be done when the gain of a node overcome the threshold.

- **Post-pruning**

  In contrast to pre-pruning, there is post-pruning. This method allows the decision tree to grow its maximum size. Then, it is followed by a tree-pruning step, which proceeds to trim the tree in something easier to analyse. Trimming can be done by replacing a sub-tree with a leaf node whose class level is determined from the majority (sub-tree replacement) or replacing the sub-tree for the most frequently used branch (sub tree raising).

3.7. Model Evaluation

The objective of model evaluation is to evaluate the performance of a learned model on a labelled test set that has not been used at any stage of model selection. This aim can be achieved by partitioning the entire set into two parts, the training data and the test data. The later will be used to know which the error rate of the model is. There are two different methods to assess the model evaluations, namely Holdout method and Cross-Validation.

3.7.1. Holdout Method

This is the most frequent strategy and it consists of dividing the dataset into two blocks in a random way, the training data and the data set. When the partition is done, the most important metric is the error rate obtained from the data test.

This method can be repeated as many times as the model requires to obtain a correct distribution of the test error rates. Furthermore, with the distribution of the error, the variance of the model can be determined.

3.7.2. Cross-Validation

This method consists of dividing the entire data into “X” equal-sized subsets (Figure 3.10) and then,

a part of subsets is used as training set and the others as a test set. It is essential to compute the test error rate and keep it. This is because when the first iteration is finished, the training and test subsets change randomly, and another test error rate appears. When all the iterations are finished, and all the test errors rates are kept, the last step is to sum all the errors and divide them for all the iterations. The aim of this method is to achieve the error's average.

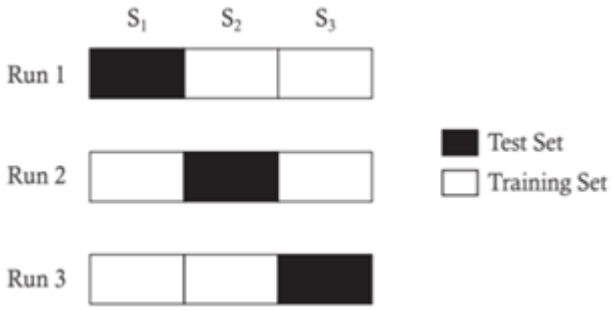


Figure 3.10: Example demonstrating the technique of X-fold cross-validation.

3.8. Presence of Hyper-parameters

The objective of hyperparameters[2] is to be determined before learning the classification model because its value can change depending on the sample. In addition, this parameter will change but it will not appear in the final classification model. It only determines which the best model to apply is.

3.9. Pitfalls of Model Selection and Evaluation

When performing model selection and model evaluation tools, some predictions must be taken into account since some pitfalls could appear. These pitfalls could mislead the final results. Some of them are easy to find and it is possible to avoid them. However, there are other pitfalls that are difficult to detect.

Two types of pitfalls are presented to understand how to handle them, in case they appear in the model.

3.9.1. Overlap between Training and Test Sets

One of the most important things when a model is applied, is not to mix the same data between the training set and test set. If it occurs, the conclusions could be wrong, and the error rate would not

[2] : A hyperparameter is a parameter whose value is set before the machine learning process begins.

be the correct one. Therefore, it is necessary to keep the data separated from the training set and the test set.

3.9.2. Use of Validation Error as Generalization Error

As exposed above, the correct error to decide whether a model is correct or not is the error obtained by the validation set. This error plays a key role during model selection, as it provides out-of-sample error estimation of models. Additionally, the data used in validation set is not the same as the one used in training set. For this reason, the validation error is a better metric than the training error to select the best models and hyper-parameters.

On the other hand, even though the validation error is the one used to select the model, the correct approach for estimating the generalization error rate of a model is to use an independent data in the test set. It is better if the data has not been used as this data will not have an influence on the final decision.

3.10. Model Comparison

Modal comparison is the step to compare results between two or more models so as to determine whether all the models have reached the same solution or not. One difficulty in comparing two models is whether the observed difference in their performance is statistically significant. The first issue is to estimate the confidence interval of models accuracy, otherwise the model is not accurately estimated. The second issue is to determine its deviation and check if it is useful.

4. Association Analysis: Basic Concepts and Algorithms

Currently, many businesses gather a great deal of data about their operations and transactions, and they need to sort and organize it to understand in which departments they can increase their profits [11].

The following provides some types of methodologies that the companies can apply to find if there are relationships that could be useful, if there are some hidden relationships… One of the main important goals of this chapter is to know which the most frequent itemsets are.

4.1. Preliminaries

Binary variables

Market basket data could represent an item as a binary variable, whose value could be 1 if the item appears on the transactions and 0, otherwise. Usually, the most common binary variables are asymmetric because the presence of an item is more important than the absence of it. This type of representation is not the best because there could be a lot of hidden parameters that may influence the final decision.

Itemset and Support Count

Let I= (a, b, c…) be the set of all the items in a business' bucket and T = (A, B, C…) be the set of transactions that are formed by the items in I. Moreover, a collection of items is called itemset and depending on the number (k) of items, the group will be called k-itemset. Note that in a transaction could be more than one itemset. For intance, one transaction could be T= (A, B, C), and two itemsets could be (a, b) and (a).

On the other hand, an important property of an itemset is its support count s(X), which refers to the number of transactions that contains a particular itemset $\alpha(x)$. An itemset will be called "frequent" if it overcomes the threshold. In the following equation, the support count depending on the total number of transactions (N) is observed:

$$s(X) = \alpha(X)N$$

Association rule

The property support count is the division between the number of times that the itemset appears and the total number of transactions. Instead, the confidence determines how frequently itemsets

$\alpha(Y)$ appear in transactions that include a specific itemset $\alpha(X)$.

$$c(X-Y) = \alpha(Y)/\alpha(X)$$

It is important to obtain support counts higher than the threshold because these nodes could be explored, and the problem could be solved. On the other hand, the confidence measures how many times a specific itemset appears in a transaction that includes a specific itemset, as well.

Formulation of the Association Rule Mining Problem

Given a set of transactions, support count and confidence of the itemstets have to be greater than minsup and minconf [3], respectively, if they are supposed to be explored as a solution. Decompose the problem into two major subtasks is one of the strategies to make the task easier.

1. **Frequent Itemset Generation**, whose objective is to find all the itemsets that satisfy the minsup thresholds.

2. **Rule Generation**, whose objective is to extract all the high confidence rules from the frequent itemsets found in the previous step.

4.2. Frequent Itemset Generation

Due to the presence a large number of candidate's itemsets together with a high computational costs would be so complexity, there are three main approaches for reducing the computational complexity of frequent itemset generation.

Reduce the number of candidate itemsets (M). The Apriori principle, described later, is an effective way to eliminate some of the candidate itemsets without counting their support values.

Reduce the number of comparisons. Instead of matching each candidate itemset against every transaction, it is possible to reduce the number of comparisons by using more advanced data structures.

Reduce the number of transactions (N). As the size of candidate itemsets increases, fewer transactions will be supported by the itemsets.

[3] : minsup and minconf are the corresponding support and confidence thresholds

4.2.1. The Apriori Principle

This theorem describes how the support measure can be used to reduce the number of candidate's itemsets explored during frequent itemset generation. The main objective is to prune all the itemsets that do not comply with the support. Thus, if an itemset is frequent, all of its subsets will be frequent, as well. By contrast, if one itemset is not frequent, the other supersets will not be either. This strategy is known as support-based pruning. Another important property is anti-monotone feature, and it proves that an itemset never will exceed the support for its subsets.

This strategy is a good option to reduce all the nodes that should be explored. At the same time, the time of computation is being reduced.

4.2.2. Candidate Generation and Pruning

The candidate-gen and candidate-prune functions generate candidate itemsets and prunes unnecessary ones by performing the following two operations, respectively:

- **Candidate Generation**. This step generates new candidate k-itemsets depending on whether the subsets in the previous iteration are frequent or not.

- **Candidate Pruning.** This step eliminates some of the candidate k-itemsets using support-based pruning. This is carried out by removing k-itemsets whose subsets are known to be infrequent in previous iterations. Thus, there will not be infrequent itemsets to explore.

Candidate Generation

Candidate generation aim to generate new candidates depending on the previous iterations and the subsets that this itemset have. Moreover, the itemset will not be a candidate if, at least, one of its subsets is infrequent.Ttherefore, this itemset will be pruned and it will not be a solution. There are few procedures that explain how to generate candidates:

- **Brute-Force Method**: this method considers every k-itemset as a potential candidate and then, depending on if they are frequent or not, this method applies the pruning step. Thus, it eliminates these candidates whose subsets are infrequent. The number of candidates that appears in this method is calculated as binomial coefficient. *K* is defined as the number of items and *d* is defined as the number of items for each itemset:

$$ç(k/d) = (4/2) = 6$$

- **$F_{k-1} \times F_1$ Method**: This method consists of joining every frequent k-itemset with another (k-1)-itemset. Note that they cannot be repeated. This option generates lower number of candidates because each of them contains at least one frequent (k-1) itemset. For intance, if X = (beans, coca) and Y=(Fanta), and both of them are frequent, they could be joined to generate a new candidate.

- **$F_{k-1} \times F_{k-1}$ Method**: This method generates new candidates from a pair of frequent (k-1) itemsets. It only occurs if their first (k-2) items are the same and they appear in lexicographic order. This method ensures that every k-itemset contains at least two frequent (k-1)-itemsets, therefore it could reduce the number of candidates.

Candidate Pruning

This operation considers every subset of k-itemset. If any of them are infrequent, this k-itemset is immediately pruned because this node will not be a solution.

4.2.3. Support Counting

This process consists in determining the frequency for every candidate itemset related to the number of transactions, that survives the candidate pruning step. The support counting of an itemset has to be higher than the minsup, if the itemset does not satisfy this requisite, it will be pruned.

One of the most interesting method to determine the support counting is to use the Hash Tree. In this method all the candidates are partitioned into different buckets and they are stored in a hash tree. During all the process, itemsets contained in each transaction are also hashed in the most appropriate buckets. The difference with other methods is, that in the hash tree, the candidates are matched with the candidates that are in the same bucket, instead of matching them with all the existing candidates.

4.2.4. Computational Complexity

The Computational Complexity is an important issue that could appear when there are a lot of candidates to explore. This problem includes the runtime of exploring all the possible solutions and the dimension of the storage. Depending on these two factors, the computation complexity

modified.

- **Support Threshold**: The support threshold is the limit that the support counts of an itemset has to overcome if it wants to be a candidate. Thus, the lower the threshold, the higher number of candidates to explore.

- **Number of Items (Dimensionality):** As more candidates, more itemsets' frequency and more computation and runtime. There are more factors that could affect the runtime and the computation, such as the number of transactions, candidate generation… Nevertheless, these are the most important ones.

4.3. Rule Generation

This section describes how to efficiently extract association rules from a given frequent itemset. Each frequent k-itemset can produce up to (2k−2) association rules, ignoring rules that have empty antecedents or consequents. Confidence-Based Pruning is one important aspect to consider is that the anti-monotone property is not described in the same way for the confidence and the support measure.

4.4. Compact Representation of Frequent Itemsets

It is useful to identify few frequent itemsets from which all other frequent itemsets can be derived. This is because is not interesting to explore all the candidates. Two examples of a compact representation are the following.

- **Maximal Frequent Itemsets:** a frequent itemset is maximal if all of their supersets are infrequent. With this method, the list of the frequent itsemsets is obtained by enumerating all the subsets of maximal frequent itemset.

- **Closed Itemset:** This compact representation consists in finding the support count of the supersets of an itemset. An itsemset is closed if none of its supersets has the same support count. If there is only one superset that has the same support count, this itemset is not closed. An itsemset will be a closed frequent itemset if it is closed and moreover its support count is greater than or equal to threshold/minsup. Figure 4.1 shows this phenomenon.

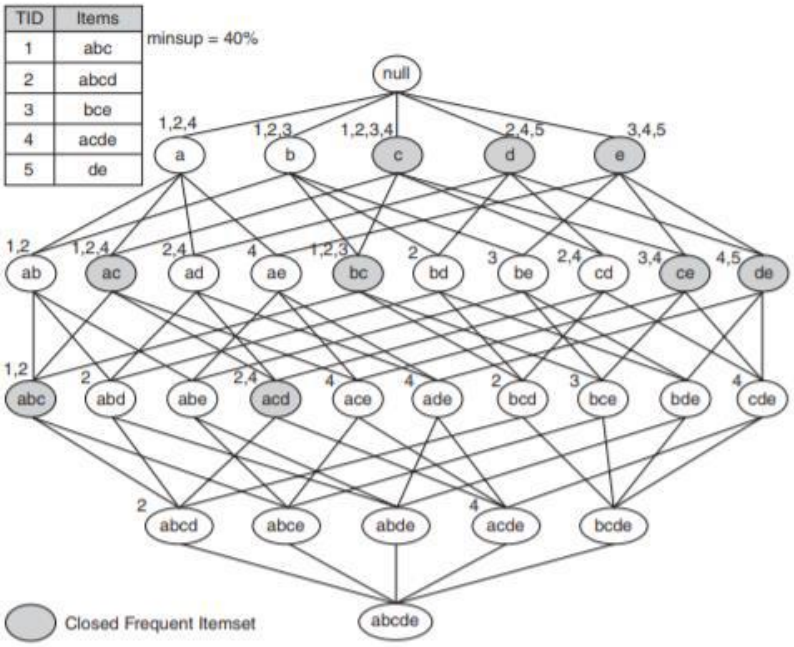


Figure 4.1: Representation of a closed itemset

4.5. Alternative Method for Generating Frequent Itemsets*

Traversal of Itemset Lattice to overcome the apriori's limitations and improve its efficiency.

Depending on the configuration of frequent itemsets in the lattice, there are strategies that are better than the others, and the lattice is traversed by different ways. There are few specific strategies:

- **General-to-Specific versus Specific-to-General:** this strategy is effective when the maximum length of a frequent itemset is not too long. If this is too long, this method is useless. On the other hand, if the problem requires a specific itemset before finding the more general, the best method is the second one. It is useful to discover maximal frequent itemsets in large transactions, where the frequent itemset border is located near the bottom of the lattice.

- **Equivalence Classes:** Another strategy is to first partition the lattice into little groups of nodes. First of all, a frequent itemset searches for some particularity in his group before moving to another. It wants to find any equivalence with the other itemsets. One way to

make equivalence classes is with the prefix or suffix labels of an itemset.

- **Breadth-First versus Depth-First**: The Depth-first approach is used to find maximal frequent itsemsets and it is advantageous to detect the frequent itemsets border. This method is faster than the breadth-first approach.

4.6. FP-Growth Algorithm

FP – Growth algorithm uses a compact data structure called FP-tree to encode all the data set. An FP-tree is a compressed representation of the input data. First of all the data set has to be read and, at the same time, the path in FP-tree has to be done by mapping each transaction. It is probable that a lot of transactions have several items in common. If it happens, the paths could be overlapped and consequently there would be more compression. On the other hand, Frequent Itemset Generation in FP-Growth Algorithm is a method to find all the frequent itemsets that end with a particular suffix, with the objective to divide the problem into smaller ones.

4.7. Evaluation of Association Patterns

The first set of criteria could be to define objective interestingness measures. These measures could be used for ranking patterns as well as for learning a way to deal with the dataset.

The second set of criteria can be established through subjective arguments. Hidden relationships could be detected by analysing the relations that could appear.

4.7.1. Objective Measures of Interestingness

This section is important to evaluate the quality of association patterns. These objective measures are always computed in a table called contingency table (Table 4.1). There are dependent and independent relationships of each pair of variables.

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

Table 4.1: Contingency's table

The support of a variable measures the probability of its occurrence, while the support s(A, B) of a pair of a variables A and B measures the probability of the two variables occurring at the same time. Hence, the joint probability P(A, B) can be expressed as the following expression:

$$P(A,B)=s(A,B)=f_{11}/N$$

If A and B are statically independent, then the expression is:

$$s(A,B)= f_{1+}/N + f_{+1}/N$$

Below, there are different factors to determine the limitations of the confidence measures.

Interest Factor (I): this factor is useful to indicate if a pair of variables are dependent or independent. Depending on the value, there are three different solutions.

- I =1 → if the variables are independent.
- I >1 → if the variables are positively related.
- I <1 → if the variables are negatively related.

Piatesky-Shapiro (PS) Measure: this factor considers the difference between the dependent and the independent relationship. Depending on the value, the solution could be:

- PS = 0 → if the variables are independent.
- PS > 0 → there is a positive relationship between the variables.
- PS < 0 → there is a negative relationship between the variables.

Correlation Analysis (ϕ): this one is the most popular technique for analysing relationships between a pair of variables. Since this factor is normalized, it only can take values from 1 to -1. Depending on the value of ϕ, the meaning is different:

- ϕ = 0 → there is no relationship between the variables.
- ϕ = 1 → it suggests a perfect positive relationship.
- ϕ = −1 → it suggests a perfect negative relationship.

IS Measure: this factor obtains a value that varies from 0 to 1. If the value is 0, there is no interaction between variables. However, if the value is 1, it exists a perfect relationship between the pair of variables.

Properties of Objective Measures

Not all the factors produce the same results and a lot of times there are some inconsistencies. Below, there are three different properties that can help to know which factor has to be applied to achieve the correct result.

- **Inversion Property**: this is a good application to know what the best factor for a specific data set is. If the binary variables are changed 1 for 0 and 0 for 1 and the result do not change, this data set is invariant and there will be factors such as correlation that would be the best one for this data set. However, if results change, the dataset is variant, and the best factor would be IS measure.

- **Scaling Property**: this property explains that a measure which is invariant do not change, despite performing row or column scaling.

Null Addition Property: a measure is invariant under the null addition when the results are not affected by increasing or reducing the f_{00}. This is applicable only when all the other frequencies in the contingency table are the same as the beginning.

5. Cluster Analysis: Basic Concepts and Algorithms

The main purpose of cluster analysis is to split data into more meaningful or useful groups. Cluster analysis is expected to capture the natural structure of the data, and thus providing a better understanding of the information contained. For intance, for marketing purposes, clustering might well be useful to segment customers into fewer number of groups [12]

Among all types of techniques, there are three which are commonly used, namely K-means, Agglomerative Hierarchical Clustering and DBSCAN.

5.1. The Basic K-means Algorithm

This is a prototype-based technique which identifies K clusters, each of which represented by its centroid. The K initial number of centroids is a user-specified parameter. Each point is assigned to the closest centroid and each collection of points assigned to a centroid is referred to as a cluster. The centroid of each cluster is recursively updated depending on the points assigned to a cluster in the immediate previous iteration. This iterative process takes place until no point changes clusters, which means, until the centroid remains unaltered.

Algorithm Basic K-means algorithm.

1: Select *K* points as initial centroids.

2: **repeat**

3: Form *K* clusters by assigning each point to its closest centroid.

4: Recompute the centroid of each cluster.

5: **until** Centroids do not change.

To proceed with this algorithm, it is necessary to define a proximity measure that represents mathematically the concept of 'closest' (step 3). Likewise, the goal of the clustering is typically expressed by an objective function. Hence, one objective function typically used is the minimization of the squared Euclidian (L_2) distance of each point to its closest centroid or, as commonly expressed, the sum of the squared error (SSE).

The cluster with the smallest squared error is always more desirable since this means that the prototypes (centroids) of this clustering are a better representation of the points in their cluster.

When it comes to selecting the points as initial centroids (step 1), performing multiple runs is advised. Each run should be done with a different set of randomly chosen initial centroids, and then the set of clusters with the minimum SSE should be selected.

5.1.1. K-means++

K-means++ is a new approach which can find a clustering solution that is optimal to within a factor of $O \log(k)$. In practice, better clustering results in terms of lower SSE can be achieved.

This technique consists of picking the first centroid at random and then picking the farthest point from the remaining centroids as a new centroid. This initialization algorithm picks centroids until k centroids have been picked. At every run, each point has a probability of being picked as the new centroid that is proportional to the square of its distance to its closest centroid.

Algorithm K-means++ initialization algorithm.

1: For the first centroid, pick one of the points at random.

2: **for** i = 1 to *number of trials* **do**

3: Compute the distance, $d(x)$, of each point to its closest centroid.

4: Assign each point a probability proportional to each point's $d(x)2$.

5: Pick new centroid from the remaining points using the weighted probabilities.

6: **end for**

5.1.2. Bisecting K-means

This algorithm is another K-means related approach which aims to obtain k clusters by dividing the set of all points into two clusters, splitting one of which, and so on, until k clusters have been found. There are different criterions to choose which cluster to split such as the largest cluster at each step, the one with the largest SSE, or even both criterions at the same time. Different choices result in different clusters.

Bisecting K-means is a technique even more efficient than K-means, and less susceptible to initialization problems.

Algorithm Bisecting K-means algorithm.

1: Initialize the list of clusters to contain the cluster consisting of all points.

2: **repeat**

3: Remove a cluster from the list of clusters.

4: *{*Perform several "trial" bisections of the chosen cluster.*}*

5: **for** *i* = 1 to *number of trials* **do**

6: Bisect the selected cluster using basic K-means.

7: **end for**

8: Select the two clusters from the bisection with the lowest total SSE.

9: Add these two clusters to the list of clusters.

10: **until** The list of clusters contains *K* clusters.

5.1.3. Strengths and weaknesses

To sum up, K-means is a simple technique that can be used for a wide variety of data types. It is also quite efficient, even though multiple runs are often performed. However, K-means is not suitable for all types of data. For instance, it cannot handle non-globular clusters or clusters of different sizes and densities. K-means also has difficulty clustering data containing outliers. In addition to these limitations, K-means is restricted to data for which there is a notion of a centre (centroid).

5.2. Agglomerative Hierarchical Clustering

There are two basic approaches for generating a hierarchical clustering:

Agglomerative: Start with the points as individual clusters and, at each step, merge (usually the closest) pair of clusters. This requires defining a notion of cluster proximity.

Divisive: Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide which cluster to split at each step and how to do the splitting.

Algorithm Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.

2: **repeat**

3: Merge the closest two clusters.

4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.

5: **until** Only one cluster remains.

The key operation of the algorithm is the computation of the proximity between two clusters, and it is the definition of cluster proximity that differentiates the various agglomerative hierarchical techniques.

MIN technique (or single link technique) defines cluster proximity as the proximity between the closest two points that are in different clusters. By contrast, MAX technique (or complete link technique) uses the proximity between the farthest two points in different clusters as the cluster proximity. The group average technique defines cluster proximity to be the average pairwise proximities of all pairs of points from different clusters. In Figure 5.1 three types are shown.

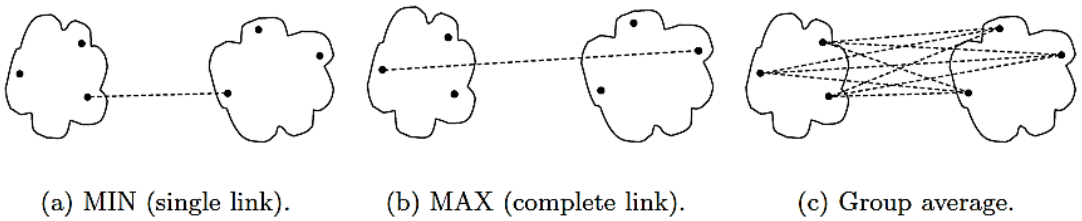


Figure 5.1: Different types of cluster proximity

5.2.1. Ward's Method and Centroid Methods

For Ward's method, the proximity between two clusters is defined as the increase in the squared error that results when two clusters are merged. Thus, this method uses the same objective function as K-means clustering.

5.2.2. The Lance-Williams Formula for Cluster Proximity

Proximity function of Lance-Williams Formula

$$p(R,Q) = \alpha_A\, p(A,Q) + \alpha_B\, p(B,Q) + \beta\, p(A,B) + \gamma\, |p(A,Q) - p(B,Q)|$$

In plain words, the proximity function of Lance-Williams Formula states that after merging clusters *A* and *B* to form cluster *R*, the proximity of the new cluster, *R*, to an existing cluster, *Q*, is a linear function of the proximities of *Q* with respect to the original clusters *A* and *B*. Thus, any hierarchical

clustering technique that can be expressed using the Lance-Williams formula does not need to keep the original data points. Instead, the proximity matrix is updated as clustering occurs.

5.3. DBSCAN

Density-based clustering locates and separates regions of high density among other regions of lower density. DBSCAN is a simple and effective density-based clustering algorithm.

5.3.1. Traditional Density: Centre-Based Approach

The first step to understand the algorithm performance is to define what is understood by the concept of density.

In the centre-based approach, density is estimated for a particular point in the data set by counting the number of points within a specified radius, *Eps*, of that point. This includes the point itself.

This method is simple to implement, but the density of any point will depend on the specified radius. For intance, if the radius is large enough, then all points will have a density of *m*, the number of points in the data set. Likewise, if the radius is too small, then all points will have a density of 1.

5.3.2. Classification of Points According to Centre-Based Density

Once defined the concept of density, it is necessary to propose an approach to determine the appropriate radius, *Eps,* for low dimensional data.

A point can be classified as being in the interior of a dense region (a core point), on the edge of a dense region (a border point), or in a sparsely occupied region (a noise or background point) as Figure 5.2 shows.

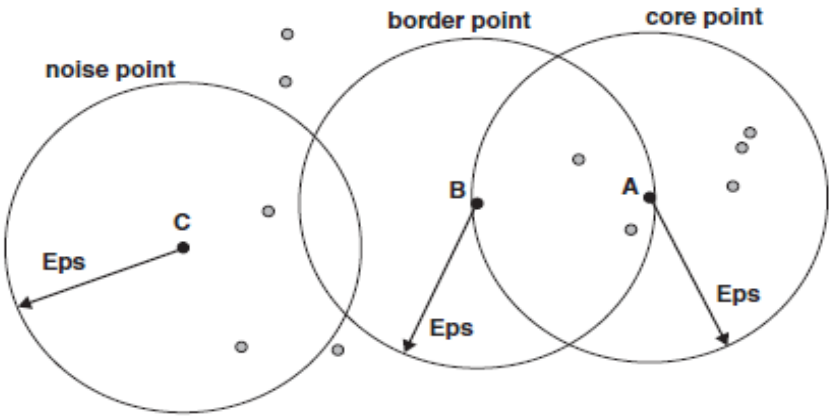


Figure 5.2: Classification of Points According to Centre-Based Density

Core points: A point is a core point if there are at least *MinPts* within a distance of *Eps*, where *MinPts* and *Eps* are user-specified parameters.

Border points: A border point is not a core point but falls within the neighbourhood of a core point. A border point can fall within the neighbourhoods of several core points.

Noise points: A noise point is any point that is neither a core point nor a border point.

5.3.3. The DBSCAN Algorithm

Any two core points that are close enough—within a distance *Eps* of one another—are put in the same cluster. Likewise, any border point that is close enough to a core point is put in the same cluster as the core point. Noise points are discarded.

Algorithm 7.5 DBSCAN algorithm.

1: Label all points as core, border, or noise points.

2: Eliminate noise points.

3: Put an edge between all core points within a distance *Eps* of each other.

4: Make each group of connected core points into a separate cluster.

5: Assign each border point to one of the clusters of its associated core points.

5.3.4. Strengths and Weaknesses

DBSCAN is relatively resistant to noise and can handle clusters of arbitrary shapes and sizes. Thus, DBSCAN can find many clusters that could not be found using K-means. However, DBSCAN has trouble when the clusters have widely varying densities. It also has trouble with high-dimensional

data because density is more difficult to define for such data.

5.4. Cluster Evaluation

Even though cluster evaluation is not a well-developed or commonly used part of cluster analysis, cluster evaluation, or cluster validation as it is more traditionally called, is important.

A proper cluster validation must deal with several important issues such as determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data; determining the correct number of clusters, evaluating how well the results of a cluster analysis fit the data *without* reference to external information (unsupervised evaluation); comparing the results of a cluster analysis to externally known results, such as externally provided class labels (supervised evaluation); and comparing two sets of clusters to determine which is better (relative evaluation).

An example of unsupervised evaluation is the SSE. Unsupervised measures are often further divided into two classes: measures of cluster cohesion, which determine how closely related the objects in a cluster are, and measures of cluster separation, which determine how distinct or well-separated a cluster is from other clusters. Unsupervised measures are often called internal indices because they use only information present in the data set.

An example of a supervised index is entropy, which measures how well cluster labels match externally supplied class labels. Supervised measures are often called external indices because they use information not present in the data set.

Relative measures are not actually a separate type of cluster evaluation measure but are instead a specific use of such measures (comparison between two sets of clusters).

6. Smart Factory's Definition

The term SMART FACTORY is not easy to understand because it includes so many technologies behind it, but in general terms, the vision of these new factories is to build an automatized and collaborative world, where the intelligent robotics combine all of its efforts in production planning, control… with people's creativity (Appendix 1). In other terms, it is a concept generated from the union of virtual and physical worlds and it tries to provide some advantages in time, quality design… Furthermore, it tries to reduce the cost because of unexpected fails or production downtime. In addition, these flexible production systems are able to respond in almost real time and it allows to optimize all the processes.

Smart Factories pretend to make a quantum jump in the development of the actual industries and achieves the goals that Industry 4.0 marked at the beginning. To adapt itself continuously and immediately on different jobs is the most important objective of these factories. The second objective would be an easy changing of the products that are being manufactured. To be adapted to the requirements of all the customers would be the last objective. Definitely, the aim of Smart Factories is to investigate new technologies that develop intelligent systems for the new factories.

Moreover, with all of these new discoveries, the chain's value increases for everyone because the suppliers, the own factory, and the customers are going to realize that all the transactions are easier than before.

On the other part, there are a lot of important areas to develop, but some of the most important parts to develop and improve are those which serves as an engine [13] (Figure 6.1)

- **Smart logistics**: to enable the flexibility on the materials' mobility and its components between storage areas and production lines. If it is possible, it should be without unguided vehicles and without additional infrastructure to facilitate the movement inside the building. Additionally, with these improvements, the product's lifecycle could be lower and consequently, the business could produce more.

- **Cooperative manipulation**: to enable the robots to manipulate all the objects that are needed for assembling operations.

- **Collaborative robotics:** to enable the cooperative work between people and robots in reduced places and shared jobs. The robot should adapt its movements so as not to collide with people and be as efficient as possible.

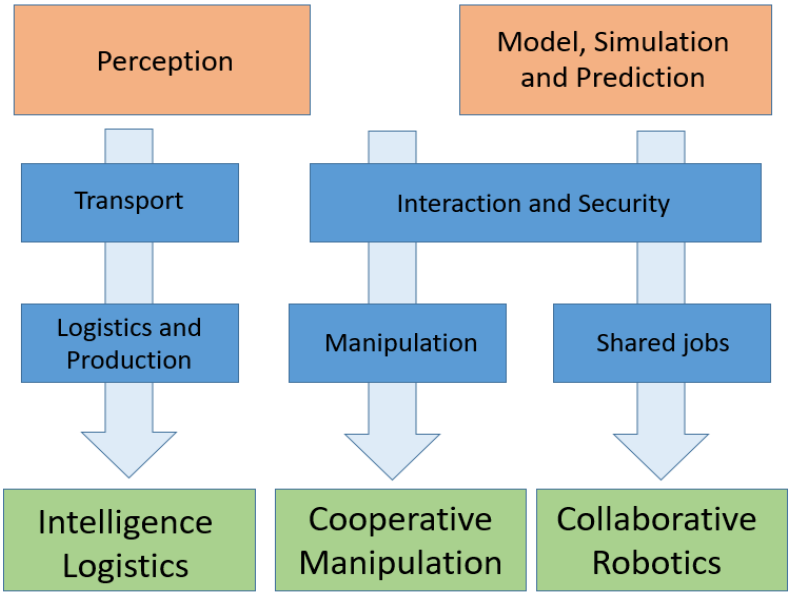


Figure 6.1: Most important areas to develop

6.1. How is the connection between everything made?

Nowadays, every new dispositive, element, terminal, sensor… is able to connect itself to the others through a physical connection or wireless. Moreover, this connection allows the interaction between them. The interaction could also be with top-ranking elements such as ERP or MES system. This is important because these systems keep all the information about the business.

The progress on technology is allowing that different machines, systems or devices are connected between them in an integrated way. This aspect is the most important one regarding who can use the information and where it can be used. All of this information will be available to be consumed by any element, from inside or outside the factory. Thanks to these advantages, mostly due to the increase in connectivity with the evolution of IT systems, better management of the information will be achieved [14] .

As a result of these changes and evolutions, the profits obtained will be a provision of information in real time. It could allow continuous improvement. Thus, this will improve the productivity, the processes' efficiency and the flexibility in production processes, among others.

7. State of the art

7.1. Deep learning for smart manufacturing

With new technologies (big data, IoT…) joined with smart manufacturing, the world of smart factory is increasing a lot. First of all, it is necessary to be focused on creating manufacturing intelligence that can provide a positive impact across all the organizations. Nowadays, the manufacturing is growing up very fast and some tools and structures are needed to manage everything, such as data collected from the product lines, manufacturing equipment, manufacturing process, labour activity… Thus, the most important thing is to analyse correctly all the data and know how to handle it to increase the volume and the gains [15] .

Regarding this article, from sensory data to manufacturing intelligence, machine learning and particularly deep learning has attracted much attention as a breakthrough of computational intelligence. One aspect to take into account is the relevance of deep learning techniques. The machines could learn and improve their skills, identify patterns and make its own decisions to determine what to achieve, as Figure 7.1 shows.

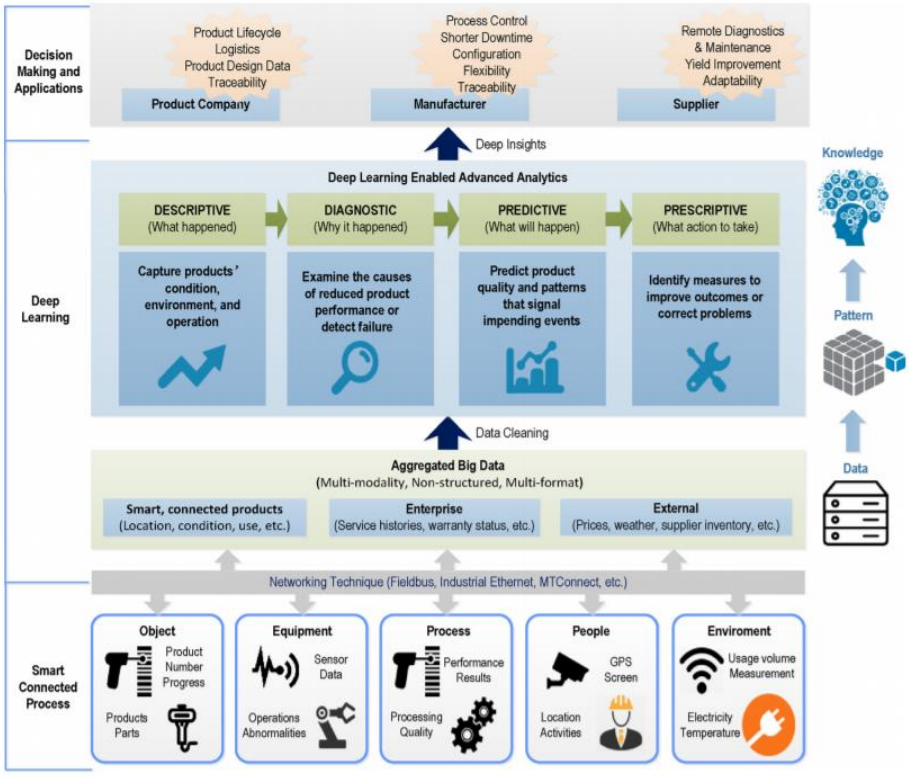


Figure 7.1: Deep learning in manufacturing.

When experts talk about data analytics, some different kinds could be identified. Some of them are described below.

- **Descriptive analytics:** this part aims to summarize what happens by capturing the product's conditions, environment, and operational parameters. When something is wrong with the product, it could be detected because the product's performance is reduced or there is any failure. Diagnostic analytics examine the root cause and report the reasons to know what to do to solve it immediately.

- **Predictive analytics:** from the available historical data, this part is able to make predictions about what will happen with future production. Furthermore, it can analyse the degradation of the equipment.

- **Prescriptive analytics:** goes beyond the other two mentioned above because it is capable to recommend some courses to improve some parts. Moreover, measures can be identified to improve production outcomes or correct the problems, showing the likely outcome of each decision.

With the analytics mentioned before, manufacturing is transformed into highly optimized smart facilities. There are some benefits such as reducing operating costs, keeping up with the needs of the costumers, improve the productivity because of the reducing downtime…

However, various deep learning architectures have been developed and they are taking a lot of value in a short time. These models are the building blocks to construct comprehensive and complex deep learning techniques. These architectures are explained below.

7.1.1. Convolutional neural network

This architecture is a multi-layer feed forward artificial neural network that it is used for two-dimensional image processing. It is also used to analyse one-dimensional sequential data including natural language and speech recognition. In this technique, the feature learning is achieved by alternating and stacking convolutional layers and doing pooling operations. These convolutional layers are mixed with raw input data and without going through any filter. Then, once the layers are created, it shows the most significant features with a fix-length by pooling operations such as max pooling and average pooling.

- **Max pooling:** this operation selects the maximum value of one region of the feature as the most significant. This one is very useful to show sparse features.

- **Average pooling:** this section determines the mean value of the region and take it as the pooling one.

Once the layers and the operations are done, the next step is to convert the two-dimensional map in one dimensional vector. Nevertheless, this vector will be used to build the model with a softmax function. Finally, the back propagation is used to train CCN by minimizing mean squared error and improving the solution.

7.1.2. Restricted Boltzmann machine and its variant

This technique consists in two-layer neural network that consists in one hidden layer and the other one is visible. It exists some symmetric connections between hidden and visible units but there is no physical connection between them within the same layer. On the other hand, the visible layer is in charge to keep the input data while the hidden one is used to extract features and its nodes are independent. The weights and offsets of these two layers have to be adjusted in order to make the output of the visible layer as similar as possible to the input.

However, the parameters of the hidden layers are treated as features because they will be used to characterize the input data to realize data coding and, if it is possible, a dimension reduction. Once this is done, is necessary to sort out the data and make a regression with the supervised learning methods as Naïve Bayes, BPNN…

Finally, this method has one advantage, because it extracts required features from training datasets automatically, the minimum value (the worst one) is avoided.

7.1.3. Auto encoder and its variants

AE is an unsupervised learning algorithm that extracts features from the input data. Moreover, this architecture does not need any type of label information to extract it, and it is formed by two parts. The first one is the encoder, and its mission is to compress the data that comes from the input of high dimensionality by mapping it to a hidden layer. The second one is called decoder and it is used to reconstruct the input data. Additionally, depending on which kind of function is adopted by the activation function, the input data will be more or less complex and difficult to analyse. Therefore, the number of hidden layers will depend on the input's data typology. Figure 7.2 shows how it works.

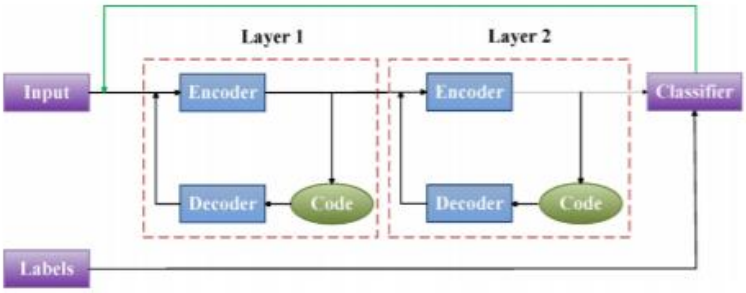


Figure 7.2: Auto encoder and its variants

Several variants could be the following ones.

- **Denoising Auto Encoder (DAE):** it is trained to reconstruct the corrupted data by adding noise and forcing the hidden layer to discover more robust features.

- **Contractive Auto Encoder (CAE):** in order to avoid small perturbations, it is necessary to learn more robust representations of the input.

7.1.4. Recurrent neural network and its variants

Comparing with traditional neural network, RNN has specific connections between the neurons that are in the layers. These connections form cycles for sequence data, for this reason, this technique is one of the best for feature learning from sequence data. RNN allows information to stay in hidden layers and it captures previous states at different time steps.

It is said that the information is in hidden layers but if the sequential input is taken as a vector, this current hidden state could be calculated by two parts through the same activation function. The first part comes to the input and the second one is obtained by the information that is in the hidden layers at the previous step. Once this is done, the output can be calculated with the current hidden state through a function.

At the end of this process, the state that was hidden is very useful because this state will be a representation of the input data and a conventional multiplayer perceptron is added on top of the map the data acquired.

On the other hand, RNN cannot deal with long-term sequence data because if it happens, the outcome could be different from the one expected. Regarding this problem, some investigations were done to find a way to solve it and it seems that the long short-term memory (LSTM) could be one of the best solutions. The idea to take into account is that LSTM is a cell state, which allows the

information of previous steps to flow down with linear interactions.

Comparing the two methods seen, the most important aspect to keep in mind is that, in the single recurrent structure in RNN, the memory is too short and it only allows to see the previous steps, while RNN extended with LSTM allows to control at any time the cell state. Finally, with LSTM, this architecture enables each recurrent unit to take the dependency that it wants whenever it wants.

Hereafter, Table 7.1 shows the main function of each technique and moreover, it adds the pros and cons that each of them provide.

Model	Principle	Pros.	Cons.
CNN	Abstracted features are learned by stacked convolutional and sapling layers	Reduced parameter number, invariance of shit, scale and distortion	High computational complexity for high hierarchical model training.
RBM	Hidden layer describes variable dependencies and connections between input or output layers as representative features	Robust to ambiguous input and training label is not required in pre-training stage	Time – consuming for joint parameter optimization
AE	Unsupervised feature learning and data dimensionality reduction are achieved through encoding.	Irrelevance in the input is eliminated, and meaningful information is preserved	Error propagation layer – by – layer and sparse representations are not guaranteed.
RNN	Temporal pattern stored in the recurrent neuronal connections and distributed hidden states for time-series data	Short-term information is retained and temporal correlations are captured in sequence data	Difficult to train the model and save the long – term dependence

Table 7.1: Summary of each technique

7.2. Machine learning applications to smart manufacturing

According to the article, nowadays the computational intelligence is one of the most important factors to improve the factory's efficiency and it is very useful to enable accurate insights for better decision making.

Machine learning can be applied in a lot of different sections when it comes to talking about smart factories and manufacturing lifecycle. Some of them are design, evaluation, production, operation, and sustainment**.** In addition, as a result of this study, fault detection, maintenance, and decision support and product quality improvement are some different categories that can be improved or rebuild with the data mining analysis.

Therefore, the evolution of smart factories especially the manufacturing is related to the advance of data modelling and analysis in manufacturing intelligence.

Then, some applications of deep learning techniques in manufacturing are discussed in order to know which the main fields are to apply it.

7.2.1. Descriptive analytics for product quality inspection

Surface integration inspection is analysed by using machine learning and image processing techniques to detect defects or to verify that every process in the supply chain is working correctly.

Regarding the progress of machine learning, it is known that it provides reliable results in many cases, but to achieve it, it is necessary to use some pre-processing approaches. Extracting representative features with expert knowledge is required to make a correct analysis. Hence, structural-based, statistical based, filter-based and model-based techniques are necessary to reach it.

However, nowadays the factories produce more than one product. It means, the requirements of them can be different and the best solution would be to apply a flexible configuration. With this kind of configuration, production from one product to another could be shift quickly. In addition, it could be possible that a new product may present complicated structure, patterns, variations and it would be difficult to detect this sudden changes with the traditional machine learning. Therefore, traditional machine learning may lead to insufficient inspection performance in complex surface or dynamic changing process.

Thus, finding these changes would be possible with deep learning, since this technique allows to

detect difficult shapes defects, random textures and new challenges could be arranged.

Some new techniques have been developed to find difficult changes and errors in manufacturing. Convolutional Neural Network, known as CNN is originally created for image analysis, is specifically well fit for automated defect identification in surface inspection. Moreover, some papers explained cases about how to implement it. One of them explained that CNN is used to perform feature extraction directly from the pixel representation of steel images and shows lower errors than traditional machine learning, as support vector machine. The results show that this architecture can be applied to all type of surfaces without depending on whether it has texture or not. Finally, this technique works better with a small dataset.

7.2.2. Diagnostic analytics for fault assessment

Diagnostic analytics is a useful application because each manufacturing system will suffer some types of failures due to tools degradation or abnormal operating conditions, caused by excessive load, fracture of some tool, overheating, corrosion… Consequently, when a failure appears while the products are being manufactured, it incurs higher operation costs and lower productivity.

Regarding diagnostic analytics for fault assessment, machine learning, specifically deep learning, is more useful because of the amount of data from smart sensory and automation system. CNN integrates feature learning and defect diagnosis in one model and it could be applied in some fields as bearing, gearbox… Moreover, it was originally created for image analysis and now, different approaches are being investigated to find a way that explains how to construct two dimensional input from time series data. About this, there is one experiment that tries to transform time series data into a matrix, and then, if it is possible, transform it into an image.

On the other hand, there is the technique called Deep Belief Network. This architecture is useful because it has the advantage of encoding high order structures by stacking multiple Restricted Boltzmann Machines. In addition, it has been used to detect wrong diagnosis of an aircraft engine, chemical process, high-speed train, wind turbine…

Furthermore, the given input data to DBN are always the pre-processed features by Teager-Kaiser energy operator or wavelet transform rather than raw data and then, DBN technique is build following a supervised layer-by-layer learning process.

Then again, there is the Auto Encoder that has been used to unsupervised feature learning. Once these features are learned, they are fed into a traditional machine learning model for training and classification and then, the probability to detect the process faults increases.

Related with Auto Encoder there is one of its variant called sparse Auto Encoder. According to this

paper [15], this could be used to learn the features of motor current signal. Partial corruption is performed on the input to improve the robustness of feature representation.

The results of all the experiments show that stacked Auto Encoder performs the best because the error rate is lower than the others. To sum up, deep learning techniques models have increased the efficiency of traditional machine learning models with engineered features such as support vector machine and BPNN in terms of classification accuracy.

7.2.3. Predictive analytics for defect prognosis

An important item to take into account is the maintenance of all the factory's machines. In order to increase the manufacturing productivity it is essential to develop and implement an intelligent maintenance strategy. The aim is to know when a machine is close to fail or simply predict when maintenance should be performed. Having a good strategy is crucial if a factory wants to reduce the maintenance cost due to some fails in the machines. Predicting the fails is always better than changing some tools that have been broken because of corrosion, degradation…

Hence, the temporal behaviour in the historical data is vital to be able to predict and know if there is something wrong. Related with machine learning, deep current neural network has demonstrated its ability to model temporal pattern. At the moment, a general neural network named long short term memory has been investigated to predict fails or defects propagation and estimate remaining useful life of mechanical components or systems.

There are some papers that explain which have been the RNN purposes to achieve it and one of them consists in knowing the rolling bearing health status through a long-term prognosis. Another one, consists in an integrated approach of CNN and bi-directional LSTM that is presented for machining tool wear prediction. Here, CNN is used to extract local features from sequential signal while bi-directional LSTM is used to capture long-term dependence for prediction.

On the other hand, stacked LSTM network enables the learning of higher level temporal features, and it was presented to predict some anomalies of space shuttle and engine. Then again, there is Deep Belief Network that is very useful in regression models. In addition, one paper says that DBN could model the complex relationship between material removal rate and chemical mechanical polishing process parameters in semiconductors manufacturing.

8. Failure analysis

8.1. Tool wear and tool life

The article “Wear and life of the tools” explains that during all the processes and activities, the tools are submitted to a lot of actions and they have to deal with different pieces and different forces. Hereafter, are listed some of the actions that can damage the tools if there is not a correct maintenance.

- Big pressures or stress actions
- High temperatures
- Chip slip through the attack surface
- Work tool slip by the mechanized surface

These conditions entail the wear of the tool and consequently, it affects negatively the tool’s life, the quality of the mechanized surface and its dimensional accuracy. Later, two different methods will be analysed depending on which kind of wear is occurring. The first one is the tool wear on the incidence surface and the second one is the tool wear on the attack surface

Usually, the tool wear is a gradual process where the tool is more weathered as the years go by. The rapidity of the tool wear depends on some inputs as the materials of the tool and the piece, the tool shape, the cutting fluid and some other parameters such as the cutting speed, the feed rate or the cutting depth. In addition, there are two types of wear depending on the region that the tool is working on, that are shown below [16] .

- **Flank wear:** it appears in the tool’s incidence surface and it is due to rubbing the tool on the surface machining. This causes adhesive or abrasive wear or it could appear because of the high temperature which affects the properties of the tool material and the piece surface.

- **Crater wear**: this kind of wear appears in the tool’s attack surface because it changes the interface geometry between the chip and tool. With this alteration, the cutting process is affected and it is not done with the maximum efficiency. The first factor that produces it is the temperature of the tool interface-chip. The second is the affinity between the material of the tool and the piece. Moreover, the factor involved in flank wear also affects crater wear.

On the other hand, there are three possible ways about tool fails depending on the processes .

- **Failure due to fracture:** this failure is due to the pressure on the tool's tip. If the pressure and the force are higher than the permitted, a fracture is produced.

- **Failure due to temperature:** it occurs when the cutting temperature is too high for the tool material. Then, it causes softening to the tip and consequently, it produces plastic deformation and the tip deviates from the edge.

- **Gradual wear:** the gradual wear of the tool's cutting edge causes loss of tool shape, reduction of cutting efficiency, wear accelerated and the final failure of the tool. This fail is similar to the temperature failure.

To sum up, the temperature is one of the inputs to take into account. It could have adverse effects on the tool life and it could affect negatively on the surface of the mechanized piece and the dimensional accuracy of cutting. Finally, some parameters such as the tool speed and the feed rate should be controlled to keep the work safe.

8.2. Machining failures

The precision machine-tool is characterized by the capacity of the machine to produce pieces with the shape and required dimensions. Thus, it means that it is very important to achieve the correct tolerances and the surface quality desired. In addition, since one machine can produce more than one type of piece, it should be necessary to respect the required dimensions of the fundamental machine elements such as flatness and straightness of the guide surfaces, alignment of the clamping surfaces… In the article "Mechanized failures", some errors and concepts are exposed to take into account and they should be known to improve the efficiency of all the processes.

8.2.1. Important concepts

There are some concepts that need to be clear to produce the piece that the factory wants. If the error rate is higher than the threshold, then the piece will not be accepted and it will be repaired or thrown. For this reason, it is crucial to know the concepts shown below to evaluate the errors [17].

- **Resolution**: this is related to the precision. The device resolution is the lower increment of the study variable that can detect the device.

- **Correction:** this is the difference between the measured value and the real one.

- **Error:** the accuracy is measured depending on the error. The error is defined as the difference between the indicated value and the real one, obtained by a patron element.

- **Accuracy and precision:** in the field of measures, these two words mean different things. While the accuracy means the difference between the correct value and the value indicates by the device, the precision indicates if the value is well-defined or not.

- **Range and scale:** it means the difference between the low value and the high one that can be measured.

- **Geometric errors:** these errors come from the beginning of the assembly. If the assembly is incorrectly mounted, then there will be some inaccuracies and they will appear to the piece because the union between tool-piece will be wrong.

- **Hysteresis error:** is produced by a deviation of the real position and the position it should actually be. It is caused by the flexibility of the system.

8.2.2. Frequent errors

Once the concepts to take into account are seen, it is important to talk about the most frequent errors between machine-tool and in the mechanized process. The errors can come from some different sources and they are explained below.

The errors that can produce some failures to the pieces or the tools are the control errors. These types of errors are caused by the mismatch of the position loop regulator, the systematic instrumentation errors and the arbitrary errors from the servo.

On the other hand, the kinematic errors can appear. These kinds of errors are induced by the relative movements of several mobile machine's components. These components should be moved with accuracy to achieve the correct result. It is frequent to find these errors while two or more guides are working at the same time because of the linear or circular interpolation.

Other errors that can appear are the ones caused by the forces and the pressure that acts during the mechanized process. These fails take the form of static deformation of the machine low load, deformation of the tool due to a high temperature between tool-piece, tool wear due to a lot of force applied on the surface or caused by years of work, instability of the materials, vibrations...

Last but not least, it should take into account the errors produced by the thermal changes. These

errors can lead to some changes in the tool shape or material shape and then, the final shape would not be correct and the customers would not want the pieces. These errors can be caused because of the heat generated by the friction between machine-tool, the heat of the refrigerant coming from the cooling system…

8.3. Experiment's example

Smart factory needs to diagnose the root cause of failures and predict the remaining useful life of mechanical system or components. As *"A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests"* explained, some of the industries require maintenance because these systems are subjected to mechanical failures due to several features such as excessive load, overheating, fracture, wear... among others. These failures always incur higher costs and lower productivity because of machine downtime. Therefore, it is crucial to develop an intelligent strategy that allows manufacturers to determine which the conditions of each system and component are and whether it is possible to work on or not. Moreover, the maintenance will be predicted by these systems [18].

In order to talk about maintenance, the three different strategies are reactive, preventive and proactive and they are described below following the information given by the article mentioned.

- **Reactive**: is the most basic approach and this strategy consists in taking action when the failure appear. In other words, assets are deliberately allowed to operate until the failures occur. Its disadvantage is that it is too difficult to anticipate the maintenance resources and, because the failure is not planned, the time and the cost are higher.

- **Preventive**: this strategy is better than reactive and it consists in performing maintenance activities after a specific period of time or amount of use based on the estimated probability that the systems fail. Furthermore, it requires more maintenance schedules to keep on work all the systems and components, compared to the reactive strategy.

- **Proactive**: this strategy is an evolution of the preventive one. It consists in knowing the operating state of the machine. The maintenance's actions are scheduled based on equipment performance or conditions, instead of time. It allows knowing when a tool, system, machine… arrives at the end of its life and consequently, this asset has to be changed for a new one.

Predicting health condition and remaining useful life based on previous and current operating conditions is done by the discipline called prognostics and health management. According to the

paper, prognostic approaches could be separated into two categories: model-based and data-driven prognostics. The first one is based on mathematical models of system behaviour deduced from physical laws or probability distribution such as methods as Wiener and Gamma processes but it also has some limitations. One of them is that it requires a depth understanding of the underlying physical processes that leads to system failures and, related to it, another limitation is that is assumed that these underlying processes follow certain probability distributions and sometimes they are not true. For this reason, data-driven has to complete it, using the data to build some predictive models through algorithms or training data.

In addition, some research has been done and it is possible to conclude that RFs (random forest) is very accurate to predict tool wear only using an experimental dataset. It is useful to predict tool wear by using cutting force, vibrations… and, moreover, the mean squared error is lower than other techniques as ANN (artificial neuronal network) or SVR (support vector regression).

8.4. Data-Driven Methods for Tool Wear Prediction

Regarding what it is explained in the article, data-driven methods are useful to predict what will be the system's response or predict some different models. One of the most useful cases related with data-driven is the tool wear topic. Notice that, this phenomenon is very critical when it comes to talking about manufacturing, especially processes such as drilling, mining and turning. For this reason, the rate of tool wear is studied extensively and it is known that its wear depends on the cutting speed and feed rate. For several years, Taylor's equation has been the method used to know an approximation of tool wear's life. Nowadays with the new technology such as machine learning and CNC machines, some other data-driven approaches to make prognosis include ANNs, decision trees and SVMs to improve the system's health management.

8.4.1. ANN

ANN, a family of computational models based on biological neural networks, is used to estimate complex relationships between inputs and outputs. Some researchers designed several effective tool wear monitoring techniques, using ANNs based on features extracted from the principles of nonlinear dynamics or some predictive modelling approach for surface roughness and tool wear for hard turning processes. The inputs of this technique include work piece hardness, cutting speed, feed rate, etc. Moreover, ANN was also used to develop a model to predict tool flank wear in end milling operations using feed-forward back propagation or to predict tool flank wear in drilling. In addition, other experts developed an online fuzzy neural network algorithm that estimates the average width flank wear and a maximum depth of crater wear, based on cutting force and acoustic

signals emissions. Therefore, experimental results have demonstrated that ANNs is one of the best techniques to predict accuracy tool wear. Some of the experiments show the following results.

- Chen and Chen developed a process tool wear prediction and the input variables were feed rate, depth of cut, and average peak cutting. 100 experiments to train the model were done and then, the model could predict tool wear with an error of 0.037 mm on average

- Paul and Varadajaran introduced a multisensory fusion model to predict tool wear in turning processes. They did a regression combining cutting force, cutting temperature and vibration signals and the experiments conclude that the coefficient of determination was 0.956 for the regression model trained by ANN

- Karayel, another researcher presented a neural network for the prediction surface roughness in turning operations. A multilayer was created to train a predictive model using the data collected from 49 cutting tests. The results showed that the absolute average error of this predictive model done with the training set, was only 2.29%

8.4.2. SVM

Some experts developed some methods made from SVM technique. One of them developed an intelligent tool breakage detection system with the SVM algorithm by monitoring cutting forces and power consumption in end milling processes. Another system was able to recognize process abnormalities in milling. Thanks to this and some other experiments, it is possible to say that SVM can be used to estimate wear progression and predicting the cutting tools efficiency.

8.4.3. Decision tree

The last data-driven method for prognosis shown in this paper is the decision trees. This method is a nonparametric supervised learning method used for classification and regression. The aim of decision trees is to create a model that predicts a variable value that is the study's objective by the rules inferred from data features. Some experiments show that decision trees approach was demonstrated to be able to make reliable inferences and decisions on tool wear classification. These experiments were made from vibration signals, acoustic emission or cutting force signals. Another experiment was ten-fold cross-validation and it evaluated the accuracy of the predictive model created by decision trees algorithm. The maximum accuracy classification was 87.5%

8.5. How could be detected tool wear?

The following section shows the methodology for data-driven prognostics for tool wear prediction using the techniques mentioned above. The input of ANNs, SVR and RFs are detailed in Table 8.1

Signal channel	Data description
Channel 1	F_X : force (N) in X dimension
:Channel 2	F_Y : force (N) in Y dimension
Channel 3	F_Z : force (N) in Z dimension
Channel 4	V_X : vibration (g) in X dimension
Channel 5	V_Y : vibration (g) in Y dimension
Channel 6	V_Z : vibration (g) in Z dimension
Channel 7	AE: acoustic emission (V)

Table 8.1: Signals obtained with the sensors

8.5.1. Tool Wear Prediction Using ANNs

ANNs technique is a model inspired by biological networks. This model is defined by three types of parameters. The first one is the interconnection pattern between different layers of neurons. The second one is the learning process for updating the weights of the interconnections. The last one is related to the activation that converts a neuron's weighted input to its output activation. The feed-forward neural network is the most popular type of ANN among all the types that exist and back-propagation is the chosen algorithm for training ANNs in conjunction with an optimization method such as gradient descent. The next figure illustrates the architecture of the FFBP ANN with a single hidden layer and it is possible to distinguish three different zones where each zone has its own neurons or unit. These three zones will be explained below.

- The first layer has input neurons which behave as buffer for distributing the extracted features from the input data. The number of neurons in this layer is the same as the

number of extracted features from the input variables. The main characteristic of this layer is that all the values are duplicated and they are sent to all neurons in the hidden layer.

- The second layer is a hidden layer. It is used to process and connect the information from the input layer to the output layer. All the values that enter in a neuron of the hidden layer are multiplied by weights between 0 and 1 and this neuron generates only one single output which is the input data of an activation function. Then, only one value will go out from each neuron. Something similar happens to the outputs of all the neurons in the hidden layer. They are multiplied by weights and generate a single value. Once this is done, this single value is also the input data of the activation function and it will convert the weighted input to the predicted output of the ANN, which is the predicted flank wear.

- The output layer has only one neuron because the solution only requires one variable.

The performance of ANNs depends on the topology or architecture and the number of neurons that are in each layer. There might be a lot of neurons depending on the problem and the solution and moreover, the training algorithm stop if the fit criterion falls below the threshold.

8.5.2. Tool Wear Prediction Using SVR.

This technique builds a hyper plane or set of hyper planes in a space which can be used for classification and regression. This technique can be formulated as a convex optimization problem if it is a linear case and it is necessary to obtain a minimize function and some restrictions to define better the problem.

If the case is not linear, the training patterns can be pre-processed by a nonlinear kernel function. First, it is important that kernel functions fulfil the Mercer's theorem. These functions provide better accuracy than the others. According to this paper, Gaussian RBF kernel is the best kernel's function in terms of tool wear prediction. In this paper, Gaussian function is used to transform the input variable and its response into a new dataset in a high-dimensional space. It is crucial to do this because the problem changes completely. Instead of having a non-linear data, the data will be linear and it will be easier to achieve the results. Figure 8.1 shows that the dataset is linearly separable by hyper plane in a Euclidean Space.

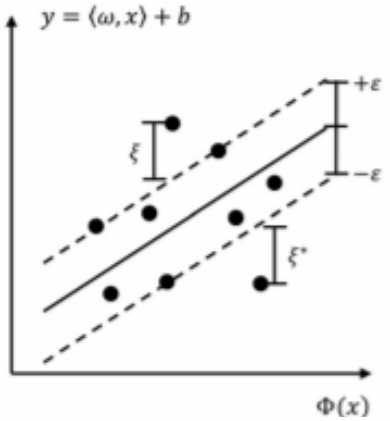


Figure 8.1: Euclidean Space

8.5.3. Tool Wear Prediction Using RFs

This technique, called random forest algorithm, is an ensemble learning method and it consists in a forest of decision trees created from bootstrap samples of the training dataset. Furthermore, if the response is continuous, the decision tree is known as a regression tree and it could be applied in the tool wear's field because this regression describes the gradual failure of cutting tools. Some of the most important concepts to take into account are described now.

- **Bootstrap Aggregating or Bagging:** Once the dataset is given, this method generates X new training models of the same size as the first one, by sampling from the original training dataset with replacement. By doing so, it is possible to observe that some observations were repeated. Moreover, Bagging is useful to reduce variance and avoid overfitting.

- **Choosing Variables to Split On:** this method says that for each bootstrap samples, it is mandatory to grow an un-pruned regression following some steps. The first one is that each node randomly samples X variables and then, the best split of this variables has to be chosen instead of taking the best split among all predictors.

- **Splitting Criterion:** it consists of dividing a partition in some regions. Then the response is modelled as a constant in each region and the splitting criterion at each node is to minimize the sum of squares. The best constant will be the average of this region. This process is repeated till the objective (predefined stopping criterion) is achieved.

- **Stopping Criterion:** one criterion that decides the complexity of the model is the tree's size. The method is a variant of splitting criterion because it consists in doing the same but it stops the algorithm when the number of records falls below the threshold.

  In summary, the steps to make a regression based on random forest algorithm are the following. The first is to create a bootstrap sample from the training data, then to build a regression tree by splitting the node into two children nodes till the stopping criterion is satisfied. The third step outputs the ensemble of trees and finally, a prediction taking into account the predictions of the trees has to be made.

  Hereafter, there is a figure that shows an experiment did through RF method. It used 500 regression trees and once the labelled training dataset was given, a bootstrap sample of size 630 was drawn by the training data. For each tree, 9 variables were selected randomly from a package of 28 and the best variable was selected among all the variables. The regression tree splits the training data into two child nodes. This process was applied recursively and it should stop if the number of records is less than the threshold imposed, which in this case,

was 5. Finally, each regression tree starts at the root node of the tree, performing a sequence of tests about the predictors and organize them in hierarchical binary structure as the Figure 8.2 and Figure 8.3 show.

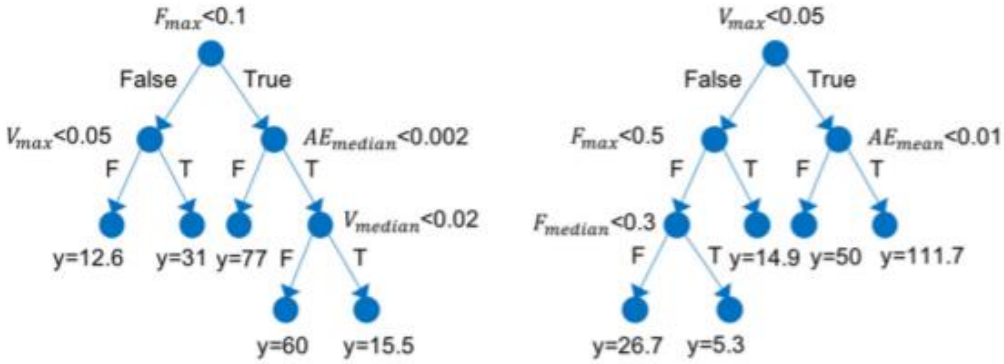


Figure 8.2: Hierarchical Binary Structure

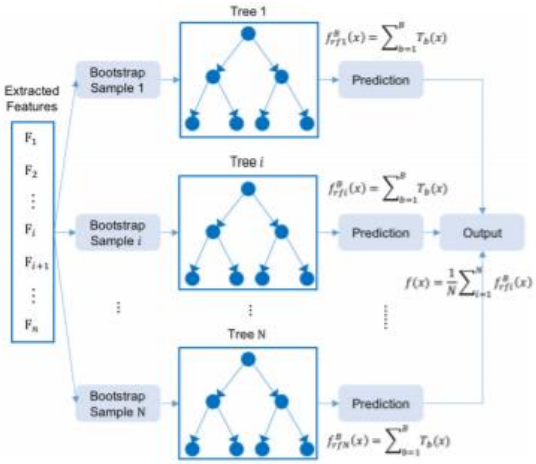


Figure 8.3: General process of Hierarchical Binary Structure

8.6. Experimental setup

In this article, all the data and information used to achieve the results come from another article titled “Fuzzy Neural Network Modelling for Tool Wear Estimation in Dry Milling Operation” written by Li.

Some of the details used in this experiment such as the cutter material and work piece material are respectively set as high-speed steel and stainless steel. Moreover, there are more variables that are very important to achieve the results and they have to be monitored. Table 8.2 shows a description of the operation condition in the dry milling operation.

Parameter	Value
Spindle speed	10.400 RPM
Feed rate	1555 mm/min
Y depth of cut	0,125 mm
Z depth of cut	0,2 mm
Sampling rate	50 kHz/channel
Material	Stainless steel

Table 8.2: Parameters of the experiment

On the other hand, to execute this experiment some cutting tests were done on different machines. Specifically, it required 315 cutting test and 3-axis high-speed CNC machine. During the experiments, some several channels as cutting force, vibration and acoustic emission data are supervised and monitored in real time to notice what happen and how it affects to tool wear. Moreover, the time of each test was 15 seconds and the three devices chosen by monitoring the different channels, were the dynamometer that was the one used to monitor cutting forces in the three different axes. The second were accelerometers and they were used to measure the vibration. The last one was an acoustic emission sensor and it was used to measure the high-frequency oscillation found within metals due to plastic deformations or crack formation. Finally, after each cutting test, the value of tool wear was measured with the objective to know how affected was the tool.

8.7. Results, Discussion and Conclusions

As it is mentioned in Section 8.6 Experimental Setup, the three variables monitored during the test were cutting forces, vibration and acoustic emission. Their data were used to find some statistical methods that can explain what happened, supported by machine learning algorithms. Regarding the data given to the system, two-thirds of the input data were selected randomly to train the machine learning algorithm. The remainder was used for model validation and it allowed to predict the results. Some sections before, three methods to predict tool wear were discussed. Therefore, this experiment has been done with each of them for comparing what was the best way to detect

tool wear. The results of prediction tool wear against real tool wear values are shown in Figure 8.4 and

Figure 8.5

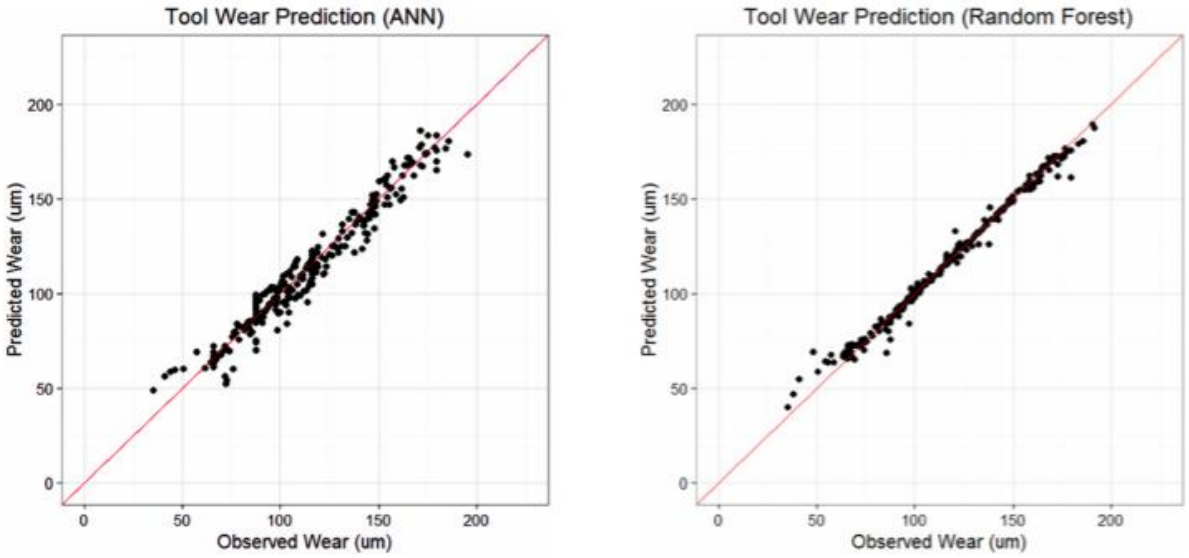


Figure 8.4: Results of ANN and Random Forest

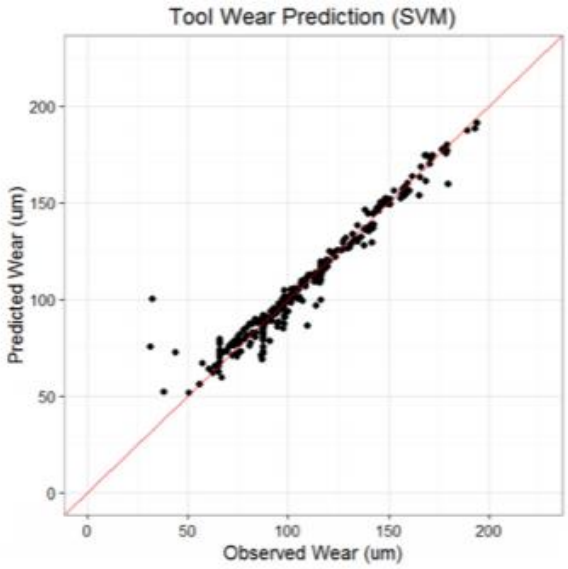


Figure 8.5: Results of SVM

In these figures, it is possible to distinguish that the most accurate method to predict the tool wear would be the random forest. This algorithm predicts the results that are the more similar to the observed ones. These statistics have been done using accuracy and training time. Regarding the accuracy, this concept is defined using the R^2, also referred to as mean squared error. It indicates the percentage of the response variable variation and it is explained by a regression model.

According to the statistics, the higher is R^2 the better, because a higher R-squared is telling the experts that more variability is being explained. In general, the higher R^2, the better regression model fits the data.

On the other hand, in this experiment, the data training used by the algorithms has been between 50 % and 90 % because the performance of ANNs model depends on the number of neurons that it has in the hidden layer. Normally, the training time increases with the number of neurons as shows Figure 8.6 and in addition, even though the prediction accuracy increases with the number of neurons, its performance does not follow the same way when there are more than 8 neurons in the layer as shows Figure 8.7

	ANN (number of neurons = 2)		
Training size (%)	MSE	R^2	Training time (s)
50	49.790	0.951	0.049
60	45.072	0.955	0.054
70	45.626	0.956	0.055
80	47.966	0.953	0.062
90	48.743	0.955	0.056

	ANN (number of neurons = 4)		
Training size (%)	MSE	R^2	Training time (s)
50	43.428	0.958	0.122
60	51.001	0.951	0.084
70	43.645	0.958	0.093
80	45.661	0.955	0.103
90	45.058	0.958	0.118

Figure 8.6: Result of ANN experiments

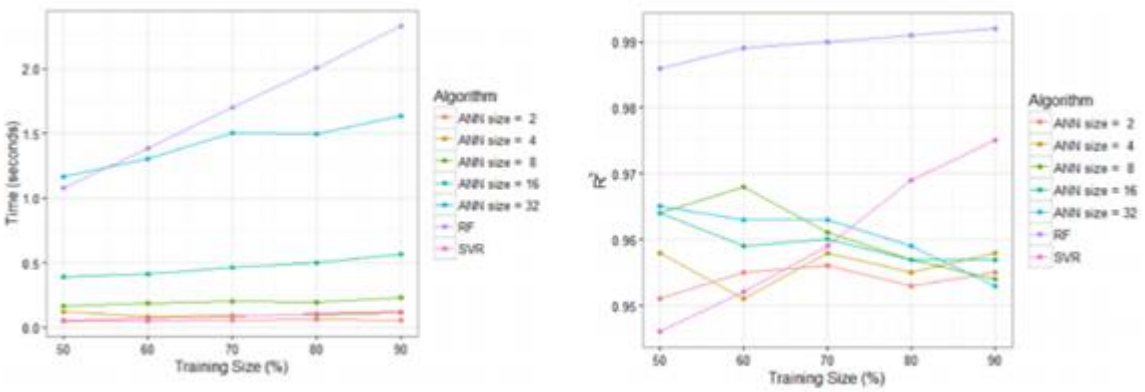


Figure 8.7: Graphical result of ANN experiments

Finally, as the previous graphics and statistics showed, the RF method seems to be the best to model the tool wear prediction, even though its time training is higher than the other. Indeed, the R^2 and MSE are better in RF method than the other two and these are the main features to select it. As shown in Figure 8.8 , comparing all training times and the performance among all the methods, RF is the one that offers more accuracy to make a good prediction.

	SVR				RFs (500 trees)		
Training size (%)	MSE	R^2	Training time (s)	Training size (%)	MSE	R^2	Training time (s)
50	54.993	0.946	0.060	50	14.170	0.986	1.079
60	49.868	0.952	0.073	60	11.053	0.989	1.386
70	41.072	0.959	0.088	70	10.156	0.990	1.700
80	31.958	0.969	0.107	80	8.633	0.991	2.003
90	23.997	0.975	0.126	90	7.674	0.992	2.325

Figure 8.8: Result of SVR and RFs experiments

9. Experimental Part with Microsoft Azure and other software.

9.1. Microsoft Azure Machine Learning Studio overview

Azure Machine Learning Studio is a collaborative, drag-and-drop tool to develop machine learning experiments, where one can build, test, and deploy predictive analytics solutions on data. []. Furthermore, Machine Learning Studio is the platform where data science, predictive analytics, cloud resources are together to solve problems.

In addition, some programming modules, as python and R-Script, are included in Azure cloud to adapt the data to the needs. Modifying the data is a key parameter and Azure has a lot of modules to transform the data.

Transforming and analysing data is the first step to build a predictive model. It is possible through data manipulation and statistical functions. After these steps, the results are generated. However, create a model requires time because this is an iterative process. Figure 9.1 shows a graphical way to understand how Azure works. Data and the own modules of Azure are combined to transform the input data. Then, the algorithms try to find the best solution exploring all the candidates and finally, the predictive result is compared with the real one.

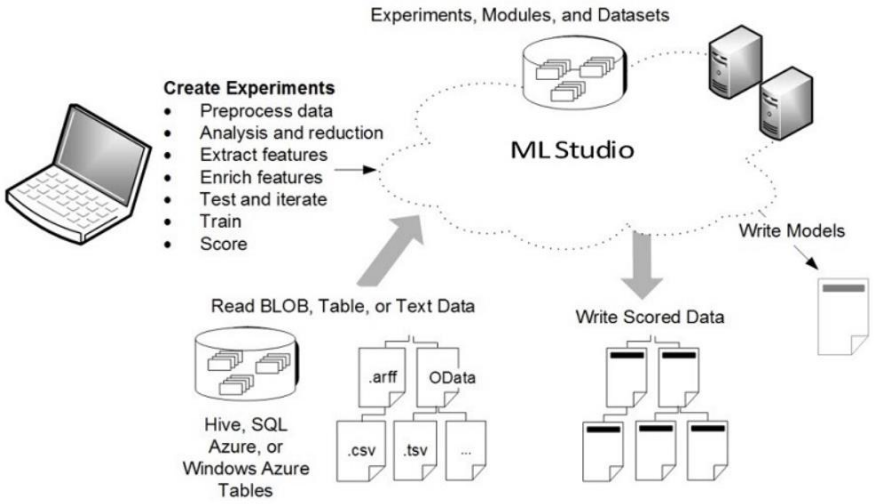


Figure 9.1: Structure of Azure platform

On the other hand, Azure Machine Learning Studio allows to work with an interactive, visual workspace to create the predictive model. Datasets and modules are dragged-and-dropped into an interactive canvas and graphically connected to form an experiment. Each module represents a set of code that can run independently and perform a machine learning task, given the required inputs.

The experiments have at least one dataset and one module. Their numbers can increase depending on the complexity of the experiment and data's transformation. Even though more than one dataset could take part in the experiment, datasets may be connected only to modules. Regarding the modules, they have inputs and outputs. The input can be formed by more than one node, but only one arrow can end in each node. Nevertheless, many arrows can come out from the output node and they represent the transference of information. Figure 9.2 shows the structure connected by arrows. While in the input only arrives one arrow, six arrows come out from the output.

Figure 9.2: Input node and output node with arrows

9.2. General Description

The data of this experiment come from certain experiments done on a milling machine under various operating conditions. Concretely, there were 16 different cases to analyse depending on the characteristics used, as they could be the material and the feed among others as shown in Table 9.1. Dta were sampled by three different types of sensors, the first was the acoustic emission sensor, the second was the vibration sensor and the last one was the current sensor. They were located at several positions to obtain the most accurate data.

Field name	Description
Case	Case number (1-16)
Run	Counter for experimental runs in each case
VB	Flank wear, measured after runs; Measurements for VB were not taken after each run
Time	Duration of the experiment (restarts for each case)
DOC	Depth of cut (does not vary for each case)
Feed	Feed (does not vary for each case)
Material	Material (does not vary for each case)
smcAC	AC spindle motor current
smcDC	DC spindle motor current
Vib_table	Table vibration
Vib_spindle	Spindle vibration
AE_table	Acoustic emission at table
AE_spindle	Acoustic emission at the spindle

Table 9.1: Data of the experiment. Description of the variables

As it has been mentioned before, there are 16 different cases with a varying number of runs as the Table 9.2 shows. The number of runs was dependent on the degree of flank wear that was measured between runs at irregular intervals up to a wear limit. Sometimes, the flank wear was not measured and so no measurement were taken.

Case	Depth of cut	Feed	material
1	1.5	0.5	Cast iron
2	0.75	0.5	Cast iron
3	0.75	0.25	Cast iron
4	1.5	0.25	Cast iron
5	1.5	0.5	Steel
6	1.5	0.25	Steel
7	0.75	0.25	Steel
8	0.75	0.5	Steel
9	1.5	0.5	Cast iron
10	1.5	0.25	Cast iron
11	0.75	0.25	Cast iron
12	0.75	0.5	Cast iron
13	0.75	0.25	Steel
14	0.75	0.5	Steel
15	1.5	0.25	Steel
16	1.5	0.5	Steel

Table 9.2: 16 different cases for the experiment

9.2.1. Experimental Setup

This experiment encompasses the spindle and the table of the Mastsuura machining center MC-510V. Two of those three sensors mentioned before (AE, Vib) are mounted on the table and the spindle of the machining center. The signals obtained by these sensors are amplified and filtered,

then fed through two RMS and finally, they are converted in data acquisition. The signal from the other sensor is fed into the computer without any treatment.

On the other hand, all the parameters necessary to develop this experiment were established by manufacturers' settings. Therefore, the cutting speed was set at 200m/min and two different depths were chosen, 1.5mm and 0.75mm. Moreover, two different feeds, 0.5mm/rev and 0.25mm/rev, were chosen. Also, two types of material, cast iron and stainless steel J45 were used. These choices equal to 8 different settings. All experiments were done a second time with the same parameters with a second set of inserts. The size of the work pieces was 483mm x 178mm x 51mm. Figure 9.3 shows how the sensors worked. Finally, Appendix 2 shows a detailed setup description.

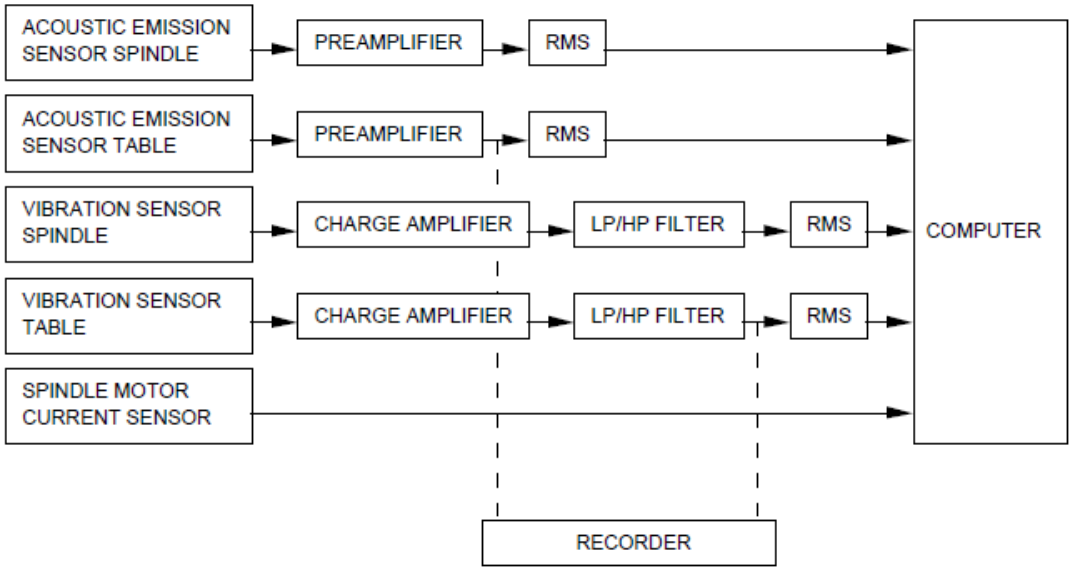


Figure 9.3: Performance of the experiment.

9.2.2. Data Acquisition and Processing

The data were sent through a high speed data acquisition board with the maximal sampling rate of 100 KHz and the sampled output of the data was used for the signal processing software. Moreover, as it is described in the previous section, some sensors signals underwent pre-processing. For this reason, the signals came from the AE sensor and vibration sensor had to be amplified to be in the correct range. Both of them were filtered by a high pass filter and after the vibration was also filtered with a low pass filter. The range of frequencies was settled between 400 Hz and 1 KHz. Finally, both signals that had to be filtered passed through an RMS device. Using it, the signal is smoother and it is more accessible to process the signal. The RMS is proportional to the energy

contents of the signal, according to the equation:

$$RMS = \sqrt{\frac{1}{\Delta T}\int_0^{\Delta T} f^2(t)\,dt}$$

Where the variables mean:

- ΔT: time constant (8.00 ms)
- f(t): signal function
- Sampling rate: 250 Hz

9.2.3. Cutting process

Talking about the interaction between the tool and the piece is very important to capture certain effects with the sensors. During the engagement of the tool in the work piece, some items have to be taken into account as could be the plastic deformation or the chip formed due to the material as shows Figure 9.4. Moreover, the energy produced affects to the radiation of heat, cutting forces, AE, etc.

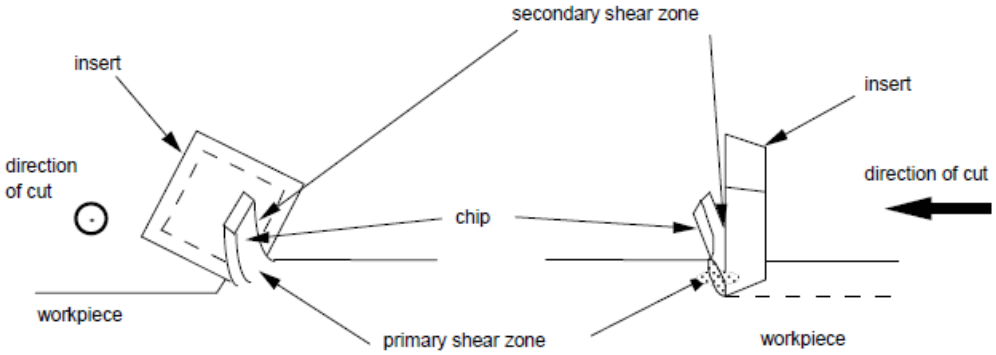


Figure 9.4: Shear zones at tool/workpiece interface

- **Acoustic emission**: is a high frequency oscillation that occurs within the metals when they are deformed or fractured. This phenomenon is caused because it releases energy and the structure of the metal is rearranged. It appears in the shear zones as it could be in Figure 9.4 and it is generated in the primary and secondary shear zone and also in the interface where is produced the interaction between tool and piece. This signal is sinusoidal and its oscillation range is between 50 kHz and some MHz. Finally, the sensor has to be located near the study zone because the signal is weaker with increasing distance.

- **Vibration emission:** this signal is a low frequency emission produced due to the acceleration of the object, because of the changes of cutting forces as a result of the variation of the of tool geometry. Its frequency range varies from 0 to 40 kHz and it happens the same as with the acoustic emission, it has to be located near the cutting zone.

The cutting forces appear in both shear zones (primary and secondary) because they are the places where the deformation takes part because of the friction between the tool and the chip, or tool with the piece. The sensors that measure these signals can be direct, that requires to install some sensors under the work piece to capture all the signals and it would be expensive, or indirect. This one is the one in charge to measure the deflection of machine parts, motor current of spindle motor or feed motors and, even though is not as accurate as a direct way, it is cheaper than it and it is easier to install.

9.3. Dataset obtained from Matlab

Before starting the experiments with Microsoft Azure and applying all the algorithms to the dataset to obtain the final results of machine learning, it is very important to understand how the data are given and how to carry out all the modifications needed to adapt the data to the machine learning program.

First of all, the dataset given cannot be opened with Microsoft Azure because it is a Matlab file and the program does not understand how the data are distributed. For this reason, the main statistical analysis and all the modifications required by the data set will be done in Matlab. Then, all the data will be adapt to the Azure program. Also, it is important to mention that the file's structure is not the best to work with, since data are given by a 1X167 structure with 13 fields as Figure 9.5 shows.

Fields	case	run	VB	time	DOC	feed	material	smcAC	smcDC	vib_table
1	1	1	0	2	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
2	1	2	NaN	4	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
3	1	3	NaN	6	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
4	1	4	0.1100	7	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
5	1	5	NaN	11	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
6	1	6	0.2000	15	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
7	1	7	0.2400	19	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
8	1	8	0.2900	22	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
9	1	9	0.2800	26	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
10	1	10	0.2900	29	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
11	1	11	0.3800	32	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
12	1	12	0.4000	35	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
13	1	13	0.4300	38	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
14	1	14	0.4500	41	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...
15	1	15	0.5000	44	1.5000	0.5000	1	9000x1 dou...	9000x1 dou...	9000x1 dou...

Figure 9.5: Structure of the dataset in Matlab

In Figure 9.5 it is possible to observe that each different case depends on several parameters, such as the feed and the material (different cases have been explained in the section that talks about the conditions of the experiment). Moreover, there are some cells that are distinguished because instead of having some values, there is a string that puts "9000 x 1". It means that in these cells, there are more than one value, concretely there are 9000 rows and 1 column and these data have to be analysed, making some transformations that will be mentioned later.

9.3.1. Is all the data useful?

There are few data to analyse and it is not possible to eliminate too many of them, because the machine learning process needs a lot of data for training and evaluating. Therefore, it would require a lot of data to make it possible and this is not the case studied. The first thing to take into account is the number of missing values that cannot be introduced in the program because it would interfere with the experiments and show wrong results. In Figure 9.5 it is possible to detect missing values because they are shown in column 'VB' with the text 'NaN'. It means that in this case, there is not any result, thus, they could not be analysed for the reason exposed before. These rows will not be eliminated now because they will be used to make statistics, but they will be removed once the Microsoft Azure starts running not to interfere the final results. Furthermore, another thing to take into account is if the 'VB' is calculated correctly or not. It means that for each case, if you are working several times with the same tool, the wear of the tool should be equal or higher than the previous run. Observing all the values of tool wear and comparing it between them, there are no mistakes detected.

On the other hand, knowing if all the sensors work correctly or not is an important step. It is crucial to realize if the tool wear was predicted well or not and the most efficient way to know it is by plotting the different values taken from the sensors installed in the cutting zone. By plotting the values of one sensor of each case, it is possible to realize that there are two cases that are not good, since the sensors show peaks that indicates something is going wrong. Concretely, these two runs can be found in case 2 run 1 and case 12 run 1. In the following figures, it is shown the difference between the sensors. In Figure 9.6, the four images represent values from 'smcDC' sensor, which has been chosen randomly from the dataset. They are similar because the sensor is working well. All the graphics show the same loop and they have 3 distinguished zones.

- The tool is getting in (set up)
- The tool works on the piece (real cutting time)

- The tool is getting out (set up)

According to these three different actions, the cutting time is only the middle zone. This zone is the most effective one.

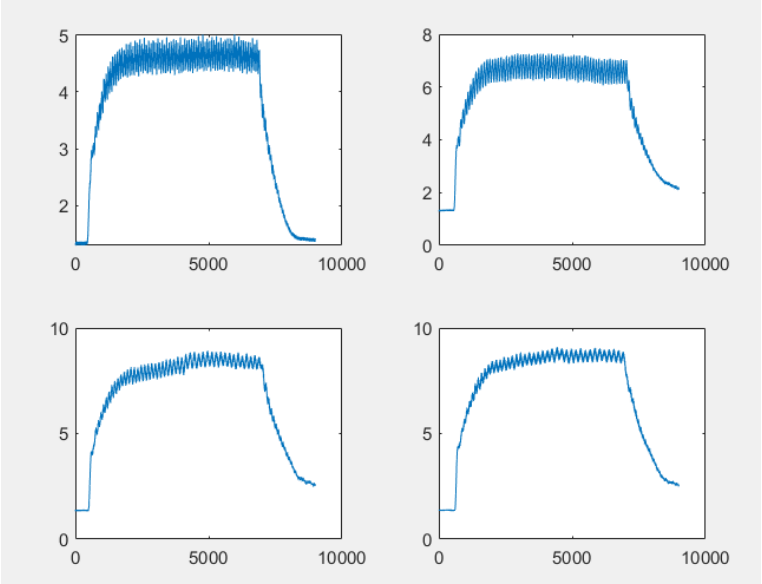


Figure 9.6: Values of smcDc from different cases

Nevertheless, as it is mentioned before, there are two runs of two different cases that are wrong because the sensors do not show the same loop as the others and moreover, it is clear that this is an error because the values are too high. In Figure 9.7 is shown the difference between two sensors that work correctly and the other two sensors that work wrong for only two runs.

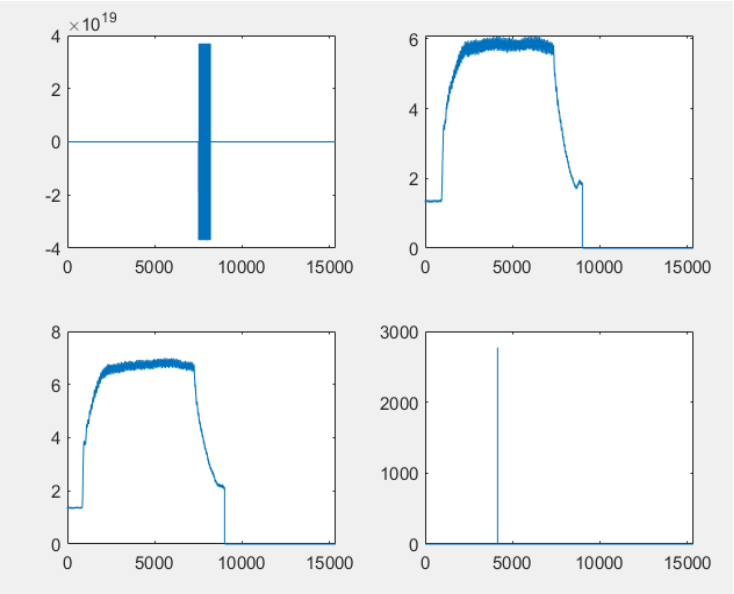


Figure 9.7: Two sensors are not working well

Once the wrong values are detected and they can be removed from the experiment, it is absolutely important to realize what happened with the time domain and if there is some need to transform the data. Analysing the dataset, each run has 9000 different values taken by the sensors and, in addition, the data have the time needed to perform each case. Knowing that the frequency was of 250 Hz (it is the same that 0.004s for each sample) and the time spent doing the entire case, it is crucial to understand that not all time has been used for the cutting process, because if it was real, there would be more than 9000 values for each case and this is not true. Thus, not all the time is used for the cutting process because there are some other needs as the setup. It means that the tool has to get in the piece and get out and these two actions consume some time.

Moreover, not all the values of each case have been taken at the same time period. One way to correct it is to transform the data to obtain a constant domain. One of the best methods to do so is the Fourier Transforms (Appendix 3), since this function is capable to decompose a function of time into the frequencies that make it up. It is also important to know that the values obtained from de FFT() are complex values. Furthermore, these values must be modified to be interpreted by the Azure program. In addition, as there are many runs for each case and every case contains its own tool wear, it would be a good decision to change the simple signals for their integral. Thereby, the cutting process will be adjusted and the tool work will be more effective than making it with simple data. Therefore, since from this moment the values of the sensors will be treated as integrals, it is important to make the cumulative sum for each run and then for every case. The experiments could

obtain more adjusted informations and the effective working would be more precise. Also, depending on the algorithms used, if some of them were 'Multiclass', it would be possible to define some boundaries with the objective to know how the tool wear is after each run.

The software used to analyse this data is Matlab, and the next images will show one of the codes created by plotting these graphics (Figure 9.8), the tool wear obtained by each run for one case (Figure 9.9) and then, the cumulative sum of tool wear for one case (Figure 9.10).

```
function sumiacum(y)
    suma=0
    i=0
    j=0
    v=[]
    [s,t]=size(y)
    for j=1:t;
        for i=1:s;
            suma=suma + y(i,j)
        end
        v=[v suma]
    plot(v)
end
```

Figure 9.8: Code created to transform data

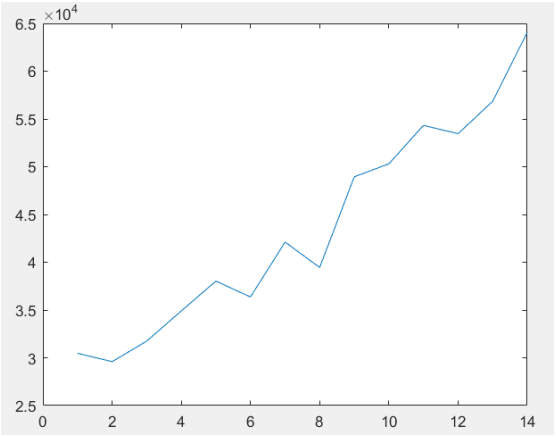


Figure 9.9: Tool wear of each run

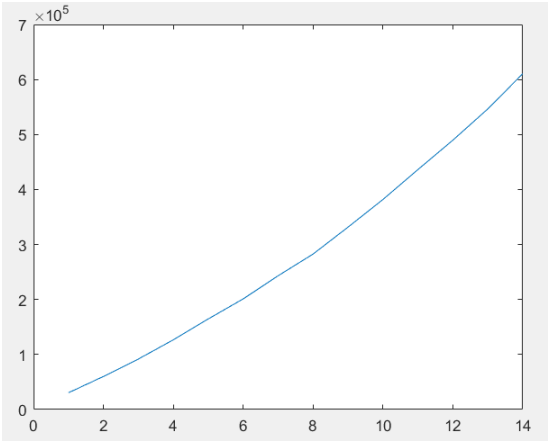


Figure 9.10: Cumulative sum of tool wear of case 1

Once the simple data have been transformed into the cumulative sum of each run and case with the objective to analyse the data better, it is the moment to explain which modifications would be the best to work out the time constraint. As it is discussed above, applying Fourier Transform is the best choice because the time is not constant. This function helps the data in order to obtain a constant domain of frequencies and amplitudes. If these changes are applied, the variables obtained by the program will change. From this moment, there will be 6x2 variables because each sensor will need

its frequency and amplitude, apart from the variables obtained before, that were the 6 cumulative sums (18 variables).

The transformation of all the sensors in a Fourier Function is easy because there is a command in Matlab called fft(), that allows you to do it. The problems appear when you try to plot Fourier values. As Fourier values are complex, treating these kinds of values is not possible. This is a bi-dimensional structure and the plot is shown in Figure 9.11. Thus, the data should be converted into one-dimensional dataset and some filters should be passed to clean the possible outliers. Applying this is tricky but with this Matlab code (Figure 9.12), it has been possible to modify it and adapt the data to make our experiments.

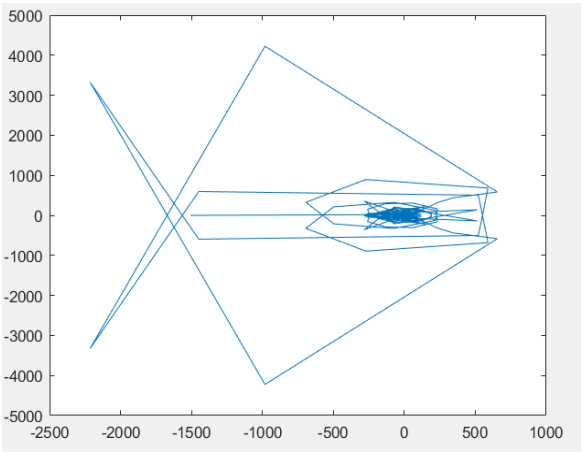


Figure 9.11: Bi-dimensional structure of frequency

```
function [ DEE ] = Marina(xy)
%función para obtener la densidad espectral de energía ( salida)
%a partir de la entrada de los sonidos de korotkoff
fm=250; %frecuencia de muestreo
duracions =length(xy);
muestrasDFT=fm*duracions;%número de muestras de la DFT.
DFT_k=fft(xy,muestrasDFT); %Transformada de Fourier discreta
DEE_k=abs(DFT_k (1:((muestrasDFT+1)/2.^2)));
DEE=DEE_k;
%graficamos los resultados
figure;
plot(DEE,'g');
title('Densidad espectral de energÍa');
xlabel('Muestras');
ylabel('EnergÍa');
end
function [DEP]= DensidadEP (xy)
%función para realizar la densidad espectral de potencia
DFT_K=fft(xy,4000);%DFT transformada de Fourier discreta
DEP=abs(DFT_K(1:2000).^2);%Densidad espectral de potencia
vect=linspace(0,125,2000);%Vector de frecuencias hasta el fs (fm/2)
%graficamos los resultados
figure
plot(vect,DEP,'m')
title('Amplitude vs Frequency')
xlabel('Frequency (Hz)')
ylabel('Amplitude')
```

Figure 9.12: Code for applying filters

So, when this code is applied, the last thing to do is to join the amplitude and the frequency to the same plot, and thereby, the two new variables are going to be together for each case and it will be easier to analyse. Notice that, in the next figures, it is possible to observe that the main frequencies are 0 or so close to this value. Finally, Figure 9.13 shows the frequency vs amplitude of the sensor 'smcAC' and Figure 9.14 shows the same data from 'vib_table'.

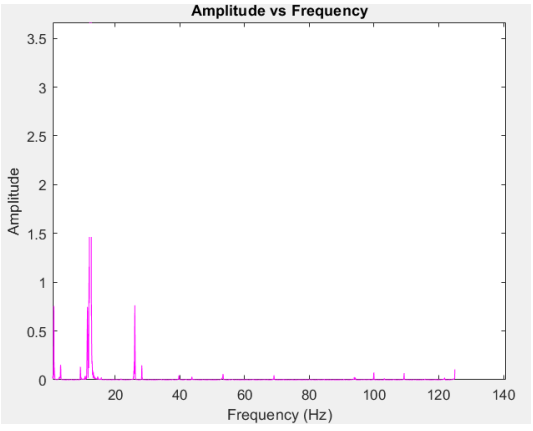


Figure 9.13: frequency of smcAC

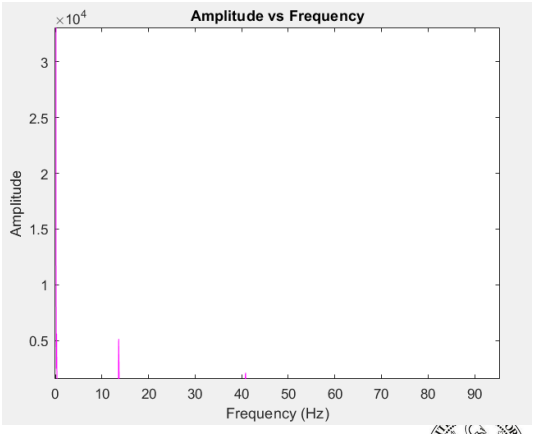


Figure 9.14: frequency of vib_table

9.4. Setup of the experiments (classification and regression)

Once the data of Matlab has been treated, the next step is to configure it because Azure does not allow all types of software. Azure is only able to understand files with specific extensions and one of them is the extension called '.CSV'. Since Matlab does not work with this type of extension, the first step will be to convert the data into CSV. The best way to perform this change is through the transfer of data into excel. First of all, when data is transferred to excel it is not in the extension Azure requires, this is why, it has to be transformed into CSV. This extension could have infinite rows but the main point is that it only requires one column. For this reason, it joins all the columns in only one, and the items are separated by commas. Figure 9.15 shows this property.

```
10,9,0.53,45,1.5,0.25,1,-1077,76650,4726.5,2500.3,2758.4,2921.3,82.216,13.5,2254.7,0,8.2586,0,1.9186,0,2.6345,0,2.7436,0
10,10,0.7,57,1.5,0.25,1,-1380.1,66627,4035.2,2643.9,2639.1,3593.2,33.607,13.667,1834.6,0,6.6101,0,2.0954,0,2.6489,0,5.5313,0
11,1,0,1,0.75,0.25,1,-1581.3,29151,8015.7,2745.2,1236.1,1560.8,14.224,12.083,309.11,0,25.609,0,2.3609,0,0.49263,0,0.77729,0
11,2,0.04,3,0.75,0.25,1,-1598.5,29212,7072.2,2713.9,1261.5,1668.9,14.73,12.083,308.18,0,19.644,0,2.2959,0,0.51116,0,0.88803,0
11,3,0.07,10,0.75,0.25,1,-1610,31173,7364.9,2610.7,1174.4,1530.2,16.876,12.111,351.03,0,20.456,0,2.117,0,0.44802,0,0.75535,0
11,4,0.07,12,0.75,0.25,1,-1606.5,33953,7117,2596,1200.8,1638.2,16.693,12.139,417.07,0,19.144,0,2.0798,0,0.46402,0,0.90392,0
11,5,0.08,14,0.75,0.25,1,-1606.4,33938,10789,2764.8,1177.9,2031.5,16.706,12.167,415.08,0,42.249,0,2.3861,0,0.44399,0,1.347,0
11,6,0.09,17,0.75,0.25,1,-1523.3,35644,9194,2616.5,1199.9,1942.4,20.696,12.167,468.25,0,31.311,0,2.108,0,0.46574,0,1.2242,0
11,7,NaN,19,0.75,0.25,1,-1548.2,37584,7196.2,2568.7,1279.1,1874.4,25.227,12.194,508.73,0,18.76,0,2.0265,0,0.53356,0,1.1622,0
11,8,0.12,21,0.75,0.25,1,-1575,39743,6548.6,2650.5,1334.4,1826.9,25.468,12.222,570.29,0,16.178,0,2.1596,0,0.58162,0,1.1028,0
11,9,0.16,27,0.75,0.25,1,-1517.6,38068,7311.7,2781.9,1304.9,1678.5,28.475,12.194,527.92,0,19.101,0,2.4088,0,0.55916,0,0.91676,0
```

Figure 9.15: Data in CSV extension

Then, the next step is to prepare the data for all the Azure's process because it will be necessary to eliminate, transform and add data. The objective is to get the best type of data to be treated for all the algorithms that could be applied. Figure 9.16 shows all the steps that the data might pass by before the algorithms treat it. In this experimental part, there will be two types of experiments depending on if the algorithms are done with classification algorithms or regression algorithms. All the steps observed in this figure are important to achieve the best results because, without them, the results would probably be wrong since they would not be treated as they need.

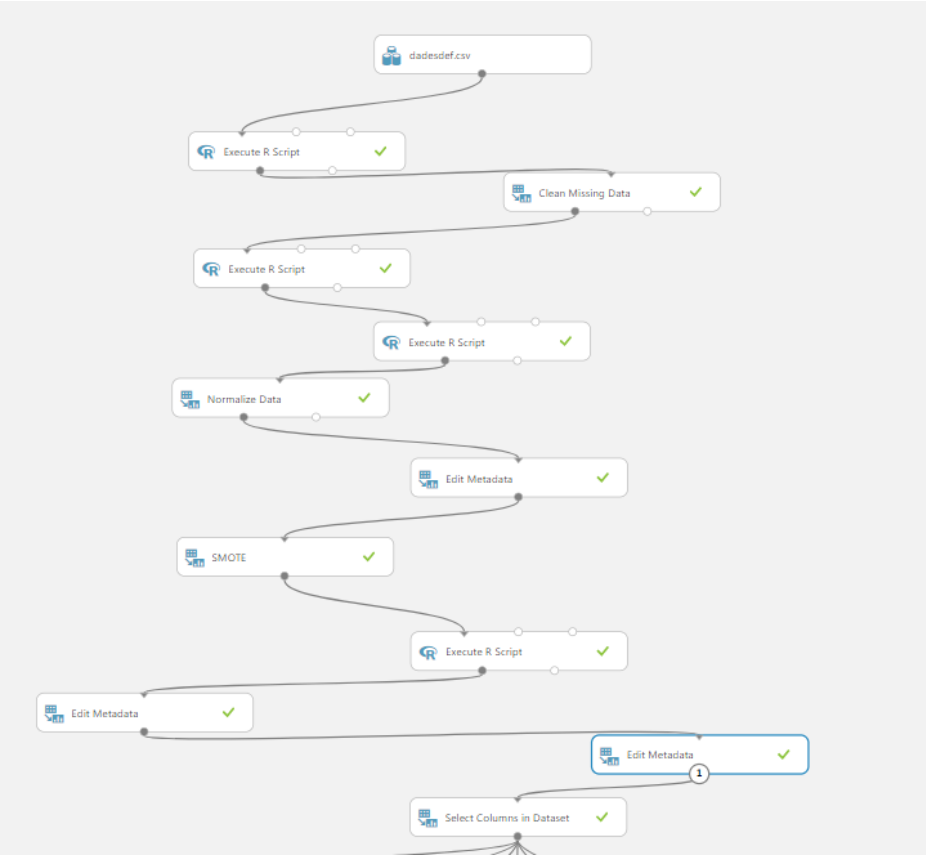


Figure 9.16: Steps for preparing the data

Execute R-Script: The first step is the Execute R-script and it transforms the data depending on the objective you want to achieve. With this step, eliminating data, transforming it, and adding new columns is easy and it allows the program to adapt correctly the data [19]. By adding R code, it is possible to perform a variety of customized tasks that are not available in the simple Azure's tasks. Creating custom data transformations, using own metrics for evaluating predictions and building models that are not implemented in Azure's modules are some of the tasks that this *Execute R-script* could perform, among others. Moreover, it is possible to realize that on the top part (Figure 9.16) there are 3 different connections. The first and the second connections are inputs and they need information also in CSV extension, while the third one has to receive a ZIP file if it wants to be read. On the other hand, on the bottom part, there are 2 connections. The one that is in the left is the output dataset and it will go to another box and the other connection is used to plot some graphs. Figure 9.17 shows the first Execute R-script's box and then, all the commands used will be explained

```r
# Map 1-based optional input ports to variables
dadesdef <- maml.mapInputPort(1) # class: data.frame

# name of the columns
colnames(dadesdef) <- c("case","run","VB","time", "DOC","feed",
    "material","smcAC","smcDC","vib_table","vib_spindle",
    "AE_table","AE_spindle", "ampli_smcAC", "freq_smcAC",
    "ampli_smcDC", "freq_smcDC", "ampli_vibtable", "freq_vibtable",
    "ampli_vibspindle", "freq_vibspindle", "ampli_AEtable", "freq_AEtable",
    "ampli_AEspindle","freq_AEspindle");

#eliminate lines
dadesdef <- dadesdef [-c(18,95),];

#adding data
dadesdef$smcAC <- ave(dadesdef$smcAC, dadesdef$case, FUN=cumsum);
dadesdef$smcDC <- ave(dadesdef$smcDC, dadesdef$case, FUN=cumsum);
dadesdef$vib_table <- ave(dadesdef$vib_table, dadesdef$case, FUN=cumsum);
dadesdef$vib_spindle <- ave(dadesdef$vib_spindle, dadesdef$case, FUN=cumsum);
dadesdef$AE_table <- ave(dadesdef$AE_table, dadesdef$case, FUN=cumsum);
dadesdef$AE_spindle <- ave(dadesdef$AE_spindle, dadesdef$case, FUN=cumsum);

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("dadesdef");
```

Figure 9.17: R-Script code to transform the data

This figure shows all the code in the first Execute R-Script box. The first line contains the port where the dataset is located. As it is possible to observe, the number is 1 because the arrow from the dataset to this module finishes in the first connection. Then, the next command allows to recognize all the columns with names, instead of being recognized with numbers. In addition, there is another command to eliminate the data that do not show correct values. This task could be done manually, but programming it allows the user to change the files whenever it is necessary. Finally, there is the section called *adding data* and these commands transform all the simple signals in its integrals. As it is possible to detect, the function FUN=cumsum creates new columns where all the values increase, in respect to the previous rows. At the end, there is the last command that indicates which is the dataset that gets out from the output.

Clean missing data: this box is the one in charge to replace missing values with the placeholder or mean, removing rows and columns that contain missing values and inferring values based on statistical methods. In this data, the most important task will be removing rows that do not contribute to the tool wear because they do not have values for it. However, they have not been removed before because the signals of the sensors are needed to create in the integral's columns.

This module has some option to clean the data, as *Replace using MICE,* that calculates a new value using some statistical methods, *Replace with mean,* that calculates the mean of the columns and replace the gap with the mean value and *Remove entire row,* that removes the entire row because the value cannot be substituted for any value, among the others. In this case, the most useful type is

removing the entire row because it is not possible to change a value with any other one, considering that these values are different depending on the material used, the time where the values have been taken, etc. Thus, all the rows that contain missing values will be removed. Mainly, the column that has more empty cells is the tool wear's column because some values were not measured.

Execute R-Script: the next step is another part of a code. This step is different depending on the type of algorithms that will be used. As it is mentioned in the previous introduction of this section, there will be two types of algorithms, some that belong to classification and the others that belong to the regression. Depending on the type of experiment that will be run, this module will be different because the program needs different inputs to solve the problem. Figure 9.18 shows the module for the classification task and as it is known, the classification algorithms need some labels to make the classification. For this reason, in this module, a new column, called *AVIS* has been created and it will be trained to improve the model. This column contains information about the operating conditions of the tool. In this code, it is seen that there are only two labels. While the wear is less than 0.3, the label will be *NO PROBLEM* which means that the tool can work without problems. But if the wear is higher than 0.3 the label will be *CHANGE*, and the tool will have to be changed. So far, there are only two labels and the reason will be exposed later, but in the end, there will be three and the one missing will be *TAKE CARE*. Furthermore, this code is divided into two parts, because it is better if the study consider the difference between the material in the tool wear's context.

```
# Map 1-based optional input ports to variables
dadesdef <- maml.mapInputPort(1) # class: data.frame

wear <- dadesdef$VB;
dadesdef$nivells <- 0;
avis <- dadesdef$nivells;
material <- dadesdef$material;
for (i in c(1:length(wear))){
    if (material[i] == 1){
        if (wear[i] < 0.3) {
            avis[i] <- "NO problem";
        } else {
            avis[i] <- "change";
        }
    }else {
        if (wear[i] < 0.3) {
            avis[i] <- "NO problem";
        } else {
            avis[i] <- "change";
        }
    }
}

dadesdef$nivells <- avis;
```

Figure 9.18: Code part of Classification

Nevertheless, there is another different R-Script for the regression part. In Figure 9.19 it is possible to observe the code for these kinds of algorithms.

```
# Map 1-based optional input ports to variables
dadesdef <- maml.mapInputPort(1) # class: data.frame

#Mandatory Library
library(plyr)

#New column for the regression
millora <- ddply(dadesdef, ~case, summarise, new = new(run));
final <- merge(dadesdef, millora, by=c("case"));

#run possible to do
final$life <- final$new - final$run;

#columns removed
final <- final[,-which(names(final) == "new")];

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("final");
```

Figure 9.19: Code part of regression

There are no similitudes between them because they need different rows and values to apply their algorithms. In this code it is possible to observe one thing that does not appear in the previous one. The command *library* is necessary because it has a crucial function to create the new column. Then, with the next two rows it is possible to create a column that indicates the maximum run per each case and with this column, it is possible to do the next step. Then, the next command creates a column in which, for each run, appears the highest number of runs that can be done before changing the tool. This column is the most important one because it is the one that will be trained and the regression will be made from its values.

Execute R-Script: this module only contains one line and the only objective is to remove all the columns that only have two different values comparing all the rows. Eliminating this is a good choice because later, the data will be normalized for each signal and if the same number appears several times in one column, then the results are exposed to be wrong because the machine learning algorithms do not work correctly.

Normalize Data: the main objective of this module is to change all the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. In this case, this module changes all the values from the signal to a 0-1 scale. This way is the best, because all the values are between 0 and 1 and, there are no significant differences between all the sensors.

Edit metadata: this module is only used to change the category of a certain dataset. Treating Boolean or numeric columns as categorical values, indicating which column contains the class label, making columns as features, among others, are some of the possible options with this module. In this experiment, the only thing that has to be transformed are the first variables of the dataset. From this moment, the variables *case, run* (regression task), *DOC, feed* and *time* will not be variables, but they will be treated as categorical values. With these changes, these columns will not be treated as random numerical numbers.

Smote (classification task): this module only appears in the setup of classification and it means *Synthetic Minority Oversampling Technique.* This is a statistical technique for increasing the number of cases in your dataset in a balanced way. The module works by generating new intance from existing minority cases that you supply as input [19] . In addition, this function does not change the number of intance of majority of cases. In this experiment, the case with the fewest intance according to the range applied was the label *CHANGE*, for this reason, it is the chosen label to improve the results. In Figure 9.20 is shown the difference between the three labels. Even though it is not a big difference, as there are few values, it is better to increase one of them to improve the accuracy and the model. Regarding one of the Execute R-script, in Figure 9.20 appears three labels and the code has only two labels. This inconsistency is made on purpose to see the initial difference between all the labels. Moreover, only SMOTE could be applied when there are two labels, for this reason, the Script only includes two types of labels.

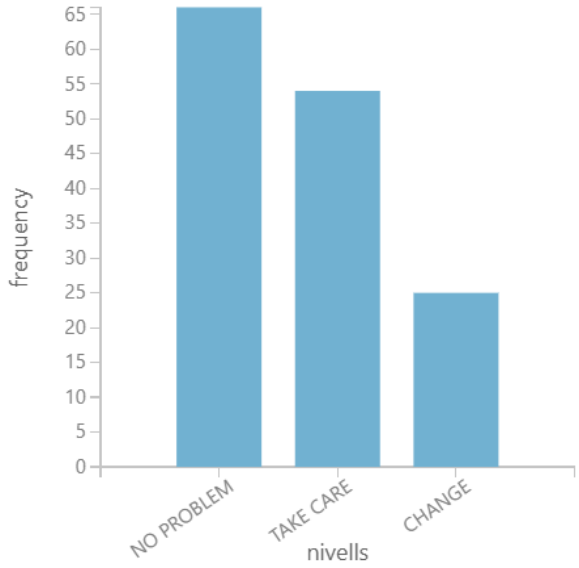


Figure 9.20: Three different labels for classification algorithms

Execute R-Script (classification task): it only appears in the classification task because it is a transformation of the other code that contains the two labels. The aim of this code is to add another label according to the ranges established looking the data. Thus, instead of having two labels, there will be three labels and the new one will be called *TAKE CARE.* This label means that the tool can work but it has to be taken into account that it can only work for few more runs. In addition, the ranges are different depending on the material of the tool and, observing the data, one of the conclusions extracted is that a lot of values of tool wear are higher than the threshold imposed to work when the run is higher than twelve. For this reason, there are some ranges for the different labels, but other condition has been added. This condition mentions that, when the run is higher than twelve, the label will be *CHANGE.* Figure 9.21 shows the code mentioned.

```
wear <- dadesdef$VB;
avis <- dadesdef$nivells;
material <- dadesdef$material;
num <- dadesdef$run;
for (i in c(1:length(wear))){
    if (material[i] == 1){
        if (wear[i]>= 0.3 && wear[i]< 0.51){
            if (num[i] <=12 ){
                avis[i] <- 'take care'
            } else {
                avis[i] <- 'change'
            }
        }
    } else {
        if (wear[i]>= 0.37 && wear[i]< 0.8){
            if(num[i] <=12){
                avis[i] <- 'take care'
            } else {
                avis[i] <- 'change'
            }
        }
    }
```

Figure 9.21: adding the TAKE CARE label

Edit metadata (classification task): the function of this module has been explained before. This module changes the category of the column called run. From this moment, this column will be treated as a categorical. This change could not be done before because the column *run* was needed to work with the script, and the variable could not be treated as categorical.

Edit metadata: its function is to change the category of the column called *NIVELLS.* From now on, this column will be treated as a label column because it will be the one that contains the predictable attribute.

Select columns in dataset: the use of this module is to choose which columns are the best for being applied in the algorithms and to exclude the columns that are not allowed. In this case, all the columns are included except VB, since it could interfere with the final results. Moreover, as VB is not the predictable attribute but it has a hard relation with trained variables, it has to be removed from the experiments.

Once the setup is configured, the next step is to define the difference between the classification and regression experiments. As it is mentioned before, classification and regression are the best kind of algorithms to apply for tool wear, but first of all, it is necessary to know how they have been trained and what the conditions to catch specific modules are.

9.5. Differences and improvements of training models

Perform good studies of a dataset is very important to know what are the best variables to take into account in order to perform all the experiments. Two other important points to be successful are the training and testing phase. Without these phases it is not possible to achieve the results and consequently, the machine learning will not learn because the program will not be able to improve any skills.

Training and testing data is mandatory to get the correct results but, there is not always a correct way to evaluate the algorithms and depending on the algorithm applied, the solution can be different from the others. First of all, it is important to mention there are many ways to train the model, but in these experiments, only two will be used. The first one is the set formed by *split data – train model – score model,* and the other one is built by *partition and sample – tune model hyperparameters – cross validate model*, as it is possible to observe in Figure 9.22 and Figure 9.23

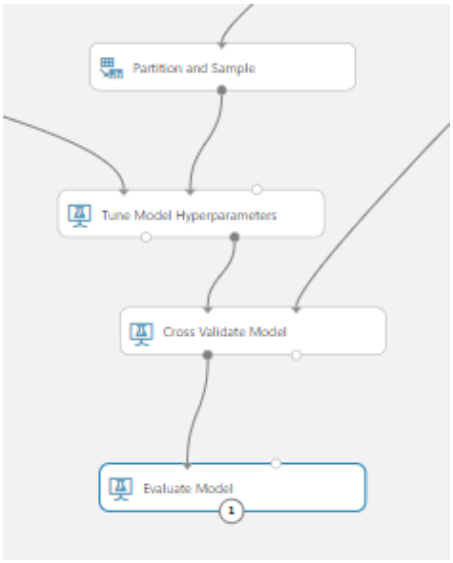


Figure 9.22: tune hyperparameters model

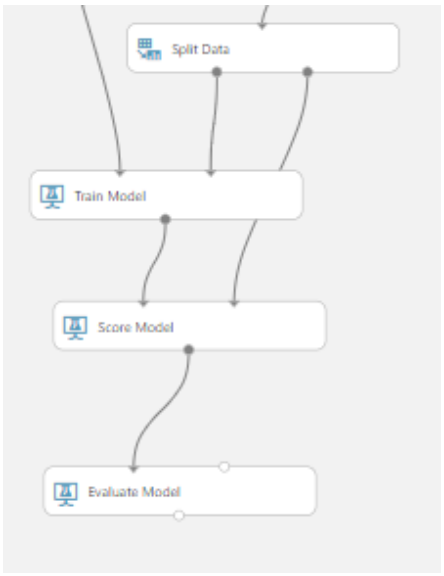


Figure 9.23: train model

9.5.1. Split data – train model – score model

This set is one of the possibilities to train the model, test it and get results to know if machine learning is successful or not, but first of all, it would be helpful to know the functions that each module has and how the parameters can change to modify the results.

Split data: this module is particularly useful when you need to separate data into training and testing sets. You can customize the way that data are divided as well. Some options support randomization of data; others are tailored for a certain data type or model type [19]. In addition, it can only create two datasets, one for training and the other one for testing. On the other hand, this module has two outputs and the arrows that get out from it cannot go wherever they want. The first output is the dataset for training data and it will go to the train model and, the other output is the test dataset and it will go to the scoring model because, in this module, the program will be able to realize if machine learning has worked out or not. However, the results can depend on the module split data, because in this module, the two datasets are divided into the fraction that the program requires. There is a cell available to write the percentage of training dataset (Figure 9.24) and the other part will be the testing set.

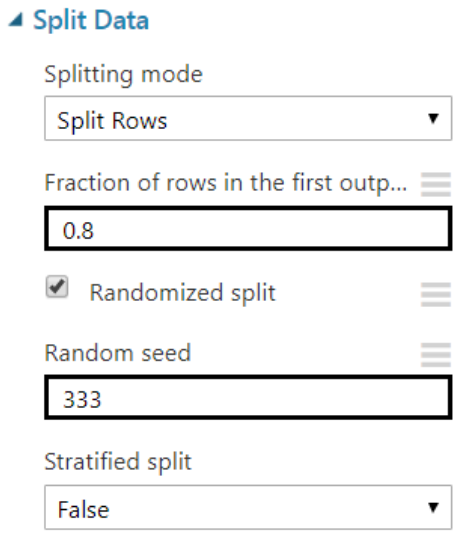


Figure 9.24: train model's parameters

Train model: train model is a classic example of supervised machine learning. It means that you must provide a dataset that contains historical data from which you can extract patterns. The data should contain both the outcome (label) you are trying to predict and the related factors (variables). The machine learning model needs the outcomes to determine the features that best predict the outcomes. In these experiments, the predictable attribute will be *NIVELLS* for classification algorithms and, *LIFE* for regression algorithms. Choosing correctly the trained attribute is crucial to

obtain the best results. Training a variable that does not give any information will not be useful for the model.

Score model: this module is the one in charge to compare the training model and the testing one. The aim of this module is to observe if the training has been done correctly. Depending on the training and the features, the results of the testing set will be more or less accurate to training data.

9.5.2. Partition and sample – tune model hyperparameters – cross validate model

Once the previous setup is done, this new type of training has to be considered because it is more complex and it supervises more solutions than the first one. Supervising more could help the model to look for more results and thanks to this, it is feasible to find the best solution.

Partition and sample: this module has several related tasks depending on the work it can do, but in this case the most accurate option is to apply split data into partitions because it is necessary to obtain some data for the training model and some data to test it. Moreover, it is mandatory to use this module if *cross validate model* is used in the experiment.

Tune model hyperparameters: this module allows to determine the optimal hyperparameters for a model. The model builds and tests multiple models using a different combination of parameters and then, all of them are compared to look for the best metrics. Moreover, it performs a parameter sweep over the specified parameter settings and defines an optimal set of hyperparameters. The process of finding the best configuration is called tuning and the two methods for finding the optimal settings are the following.

- **Integrate train and tune:** this option consists in to configure a set of parameters to use, and then let the model iterate over multiple combinations till it finds the best model. Depending on how long the process tuning is defined, the model will explore a different numbers of combinations. If the process was shorter, then it would be possible to establish a grid of parameters to finish before.

- **Cross validation with tuning:** with this option, the data is divided in some folds and each fold is tested to find a possible solution. This one is the best option because it provides the best accuracy and can help to find problems with the dataset. Applying this option is the best according to the results, but not in terms of time. This module provides better solutions but it takes more time to explore all the candidates.

In Figure 9.25 it is possible to observe the configuration of this module. All the cells in this figure are important because they need to be configured with the correct parameters. The first one needs to

be a random sweep because the model does not know which variables are the best to explore, thus it needs to find the best solution among all of them. Then, it is necessary to select the attribute to train because the model has to learn to improve its statistics. Last but not least, the model is trying to improve its accuracy and the mean absolute error has to be low. If these parameters are correct, it means that the model is working well.

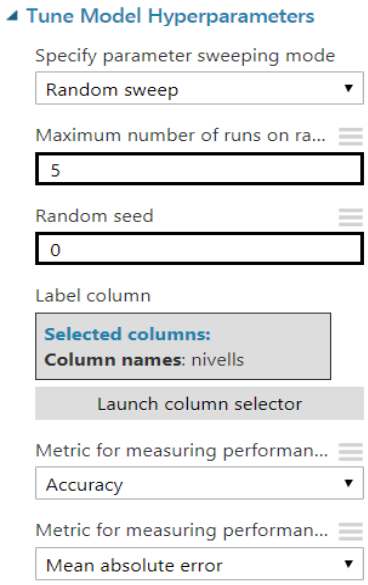


Figure 9.25: tune model's parameters

- **Cross validate model:** it takes as inputs a labelled dataset and an untrained classification or regression model. It divides the dataset into some number of folds, normally the folds are decided in the module *partition and sample*, and then it returns a set of accuracy statistics for each fold. When interpreting all the statistics of the different methods, it is possible to know the quality of the data.

Regarding the differences shown between these two types of sets, it seems that the second one will be better than the other because it explores more nodes and it makes more combinations to get the optimal one. In fact, both configurations might be applied to find which one is better and to know if the differences on the results are high or not. In this case, the variable *computational time* is not significant because the dataset is small, but if the dataset were higher, then this variable should be considered because it could affect the model negatively.

9.6. Classification

In the previous section, the setup for all the experiments and the differences between each of them have been observed. In the classification task, it will be seen which algorithms are the best for

applying and if there are some differences in the results, depending on the chosen modules.

As classification task has a lot of algorithms to apply, not all of them are useful to solve the tool wear's problem. In order to observe and research all of them, four algorithms have been chosen to get the most accurate results. Most accurate results mean that Azure's studio has to train well its dataset, thus, machine learning will be useful to predict future results.

Multiclass decision forest, multiclass decision jungle, multiclass logistic regression and multiclass neural network are the four algorithms chosen to decide which one is the best to improve the model. Appendix 4 shows the structure of all the model through Azure's platform.

9.6.1. Multiclass decision forest

The algorithm works by building multiple decision trees and then voting on the most popular output class. Voting is a form of aggregation, in which each tree in a classification decision forest outputs a non-normalized frequency histogram of labels. The aggregation process sums these histograms and normalizes the result to get the "probabilities" for each label. The trees that have high prediction confidence have a greater weight in the final decision of the ensemble [19].

These experiments have been done with the different methods explained in section *Differences and improvements of training models* and the results are shown in the next figures.

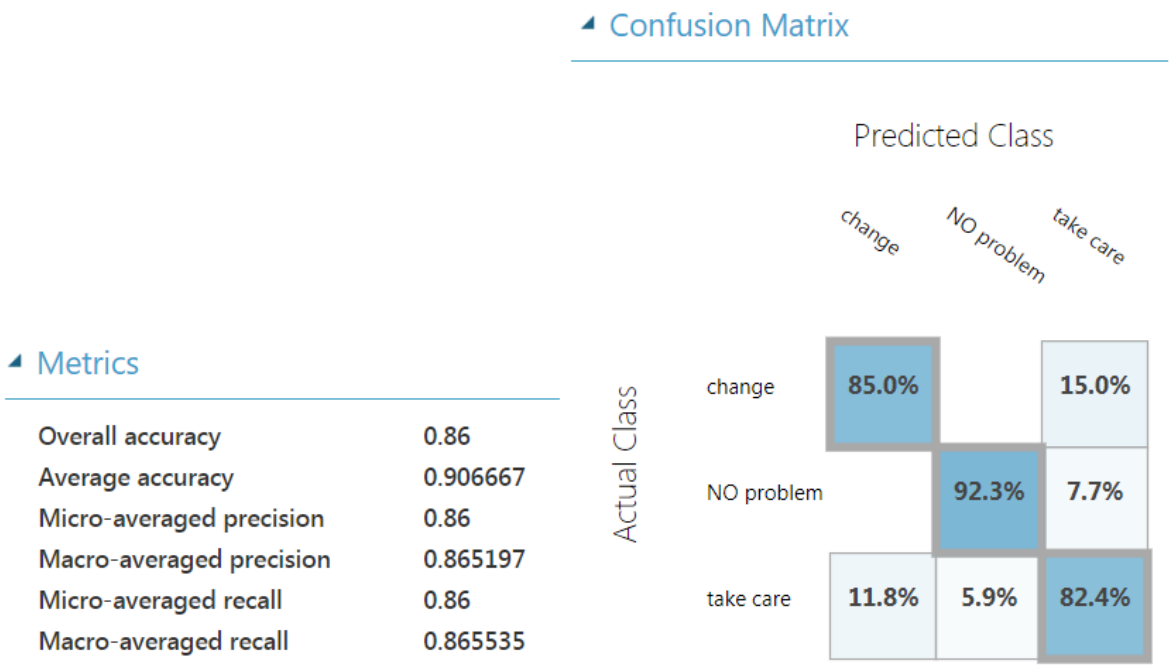


Figure 9.26: Results obtained with train model

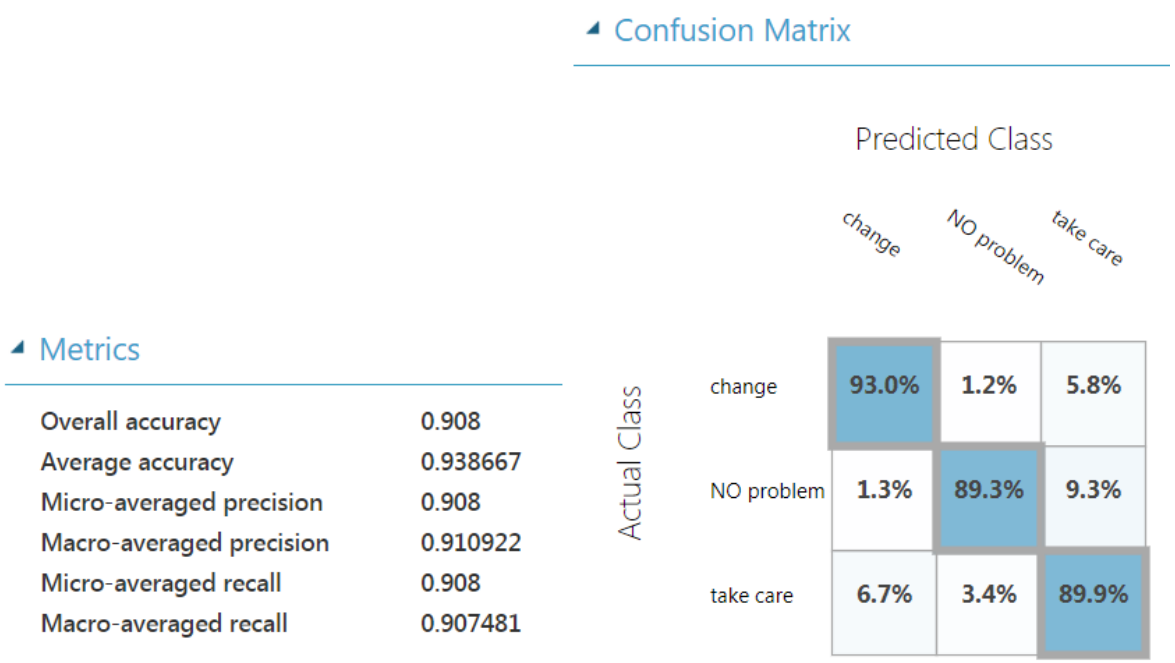


Figure 9.27: Results obtained with tune hyperparameters

In Figure 9.26 it is possible to observe the metrics and the confusion matrix of the set formed by the train model, while in Figure 9.27 the results come from the cross-validate type. The metrics explain how good the model is because they work with accuracy and precision while the matrix confusion shows the percentages of the cases that have been successful and the cases that have not. The methods are so similar but according to the results, observing the confusion matrix and the accuracy, the two critical labels are *take care* and *change* because they indicate when the tool is close to arrive at the end of its life. If it happens, it could not work in good conditions and it should be replaced. Therefore, the percentages and the accuracy of the *tune hyperparameters* are higher and it could be assumed that this method of training is better for this type of algorithm. The accuracy and the percentages are higher than 90%, thus, it could be considered as correct for this study.

9.6.2. Multiclass decision jungle

This algorithm is an extension to decision forest and it consists of an ensemble of decisions directed acyclic graphs. Some of the advantages of using it are the following. It has a lower footprint memory and a better generalization performance than decision tree, they are non-parametric models, and thus they can represent non-linear decision boundaries. Furthermore, they perform integrated features selection and classification and are resilient against noisy features. As in decision forest, these experiments also have been done with both kinds of training.

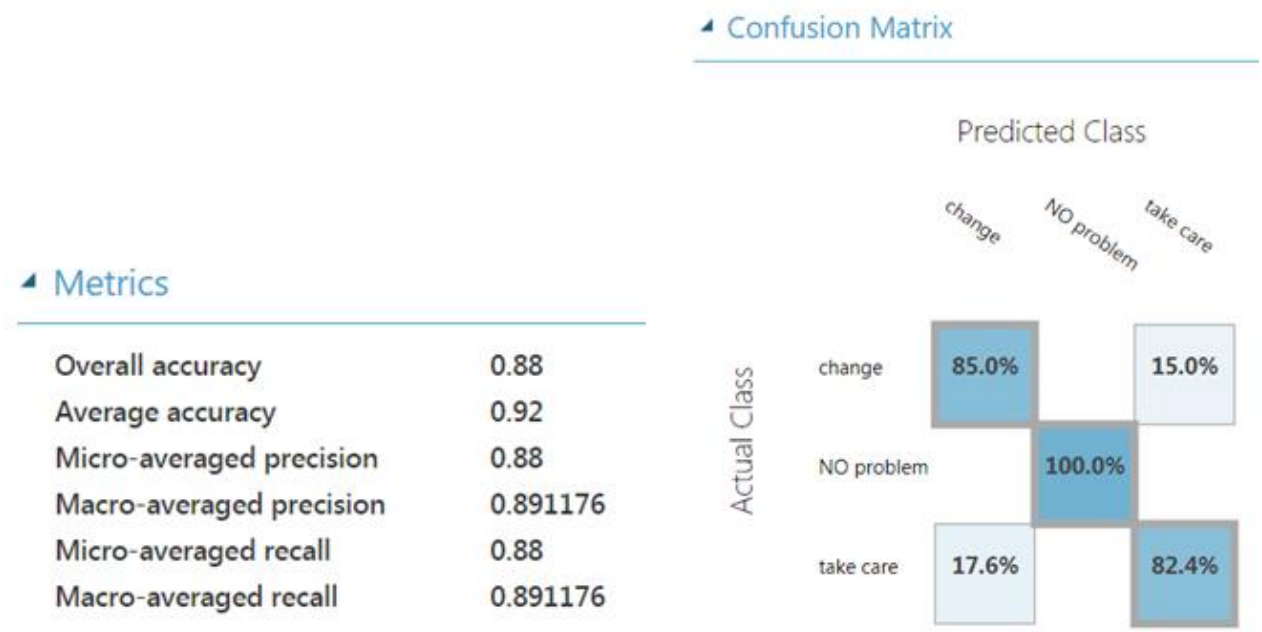


Figure 9.28: Results obtained with train model

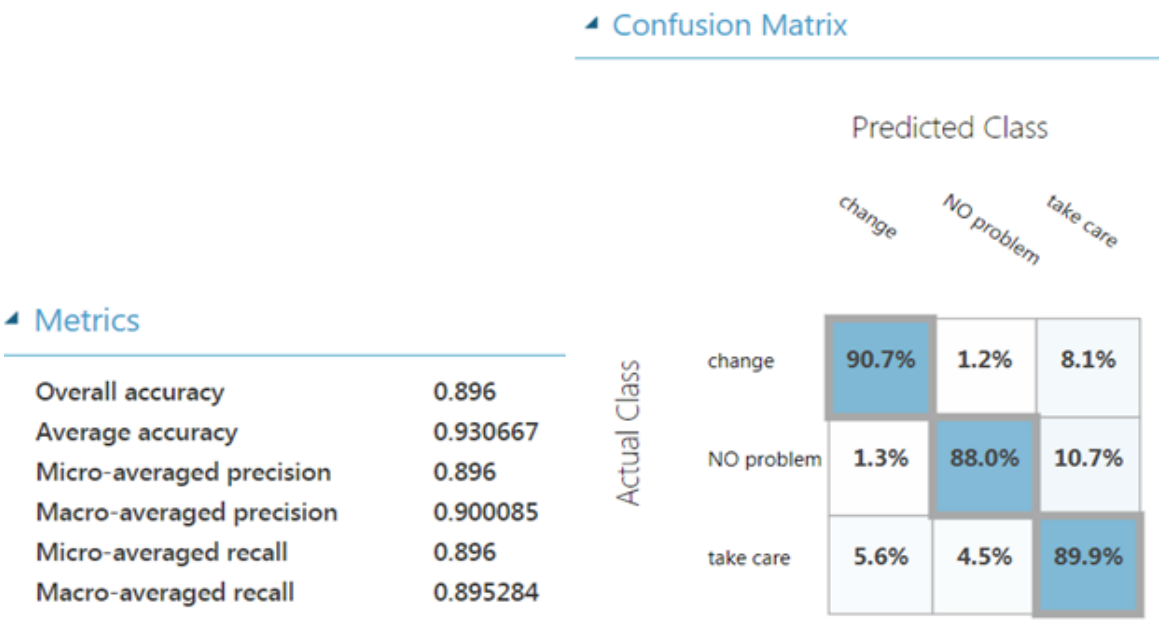


Figure 9.29: Results obtained with tune hyperparameters

In Figure 9.28 it is possible to observe the metrics and the confusion matrix of the set formed by train model, while in Figure 9.29 the results come from the cross-validate type. The results are similar to the decision forest, because this method is the same with some variations. For this reason, the model trained with tune hyperparameters is better than the other one. Exploring more combinations allow the model to find a better solution. The accuracy is around 90% and it means that it is working correctly.

9.6.3. Multiclass logistic regression

Logistic regression is a well-known method in statistics that is used to predict the probability of an outcome and is particularly popular for classification tasks. The algorithm predicts the probability of occurrence of an event by fitting data to a logistic function. Moreover it is a supervised learning method and it needs a dataset to be trained. As the other two methods, these experiments have been done with both types of training.

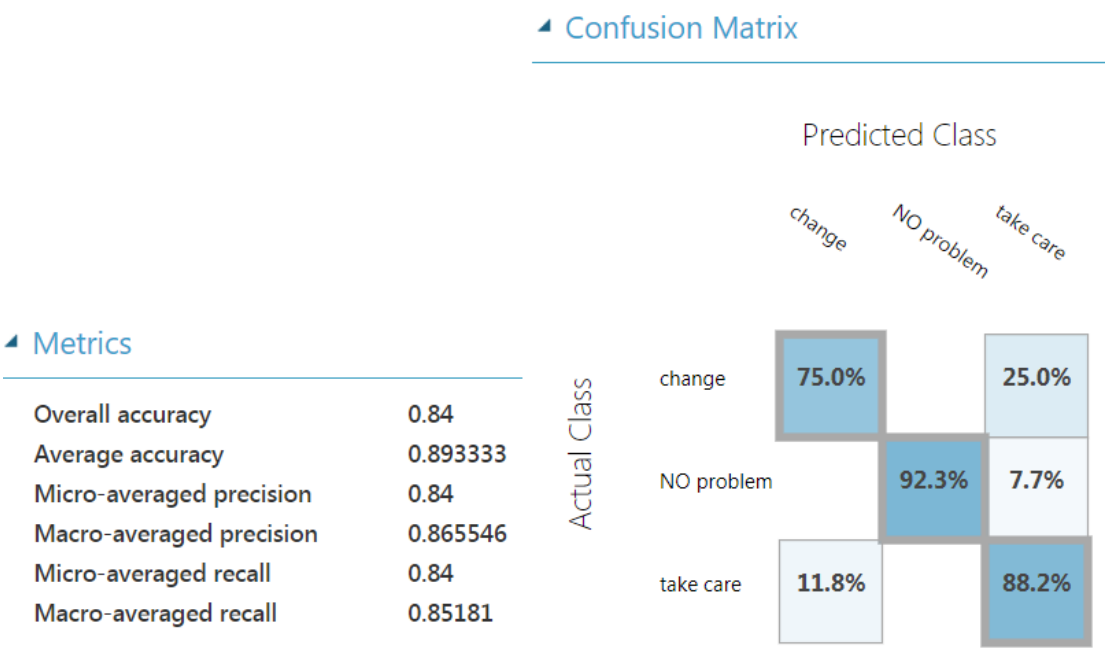


Figure 9.30: Results obtained with train model

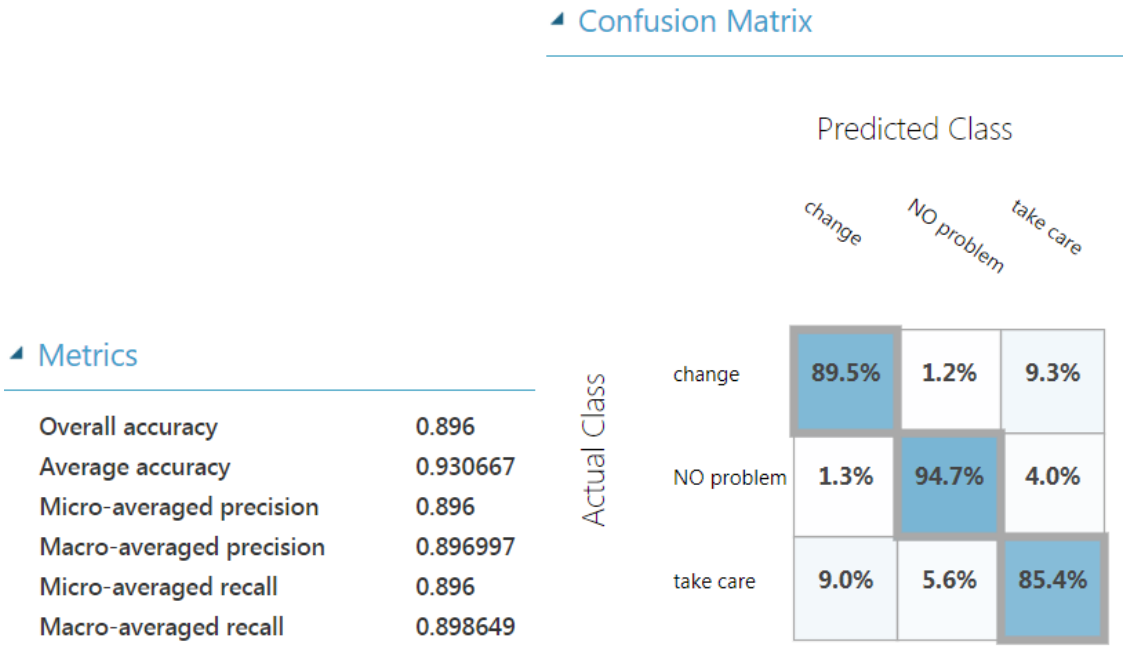


Figure 9.31: Results obtained with tune hyperparameters

In Figure 9.30 it is possible to observe the metrics and the confusion matrix of the set formed by train model, while in Figure 9.31 the results come from the cross-validate type. The difference between this algorithm and the other two is that the configuration with train model is worse because it is only successful in 75% of cases when it has to assume that the tool has to be replaced. This statistic is bad, because in 25% of the cases, the tool will be working in bad conditions and the pieces will be badly finished due to the operating condition of the tool.

In addition, the complex configuration is better than the other one but is worse than the two algorithms shown before.

9.6.4. Multiclass neural network

This algorithm is a set of interconnected layers. First of all, the inputs are the first layer and they are connected to the output layer, where the solution will be extract. In addition, in a lot of cases there are hidden layers between the input and the output. In most of the cases with few layers the model works correctly, but if the problem is very complex, it is shown that with many layers it could work better.

The relationship between the input and the output layers is learned from training the data on the input. The information only can go in one direction and always, the information travels from the input to the hidden layers and then through the output. Finally, in the output, there is an activation function that calculates the weighted sum of the values of all the nodes in the output and it gives

only one solution. This method, as the others, has also been trained with both types of training.

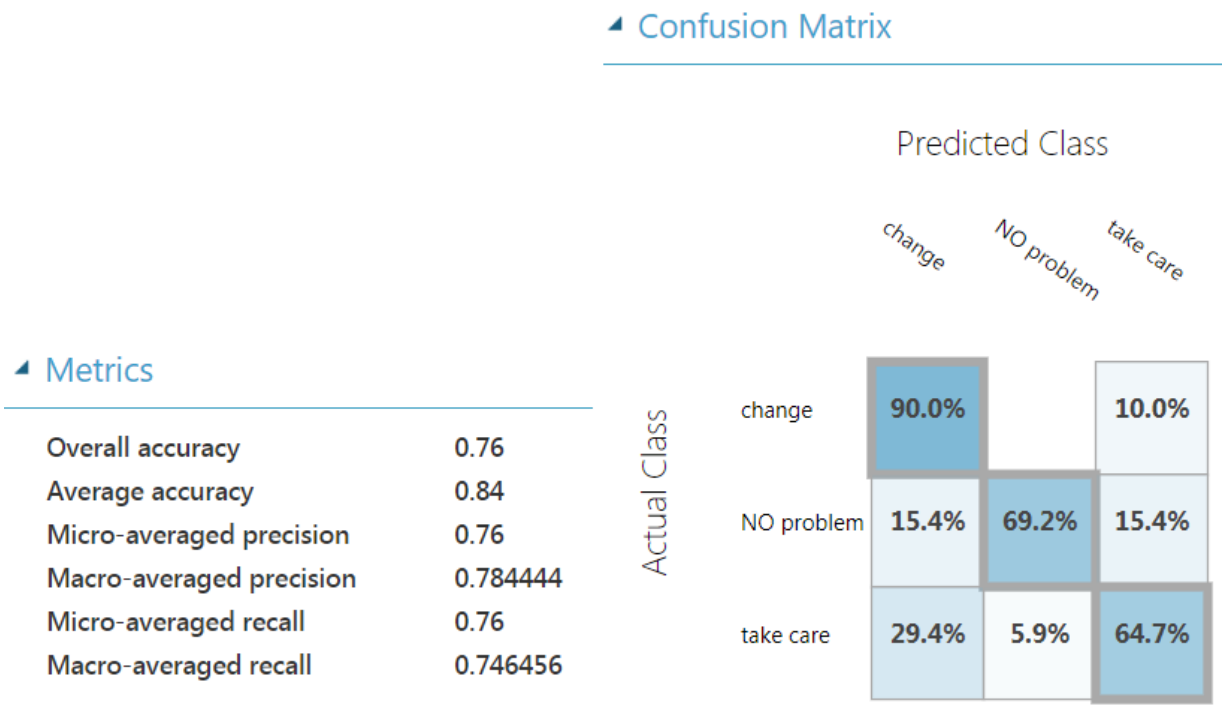


Figure 9.32: Results obtained with train model

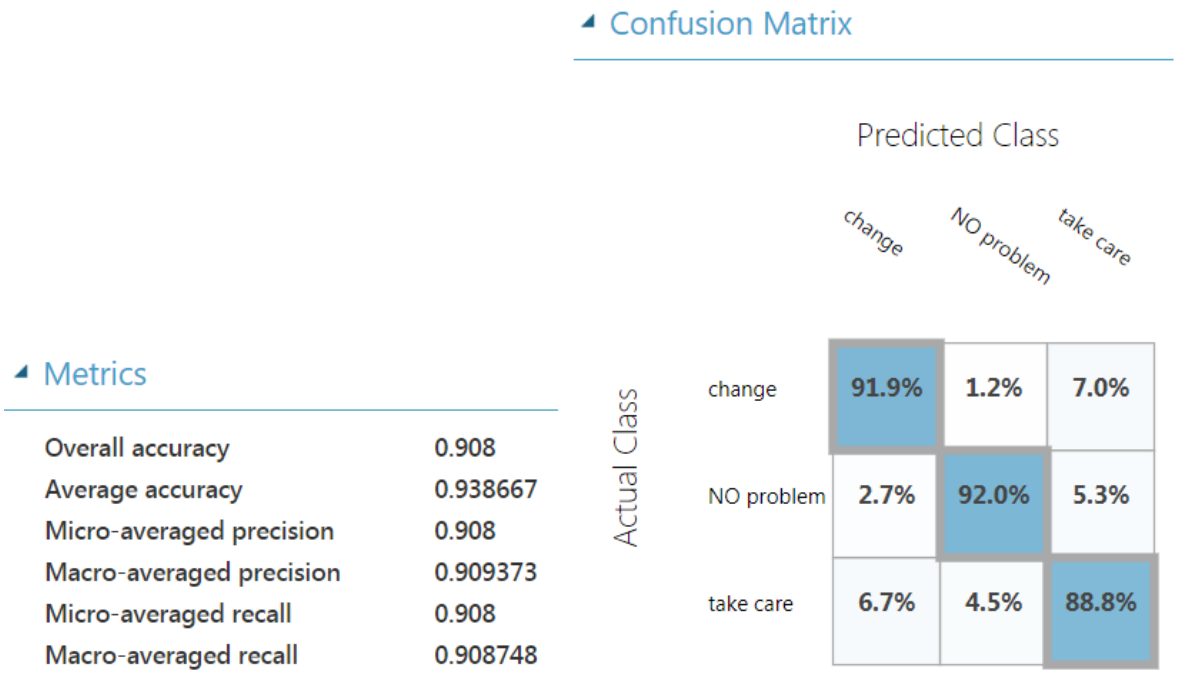


Figure 9.33: Results obtained with tune hyperparameters

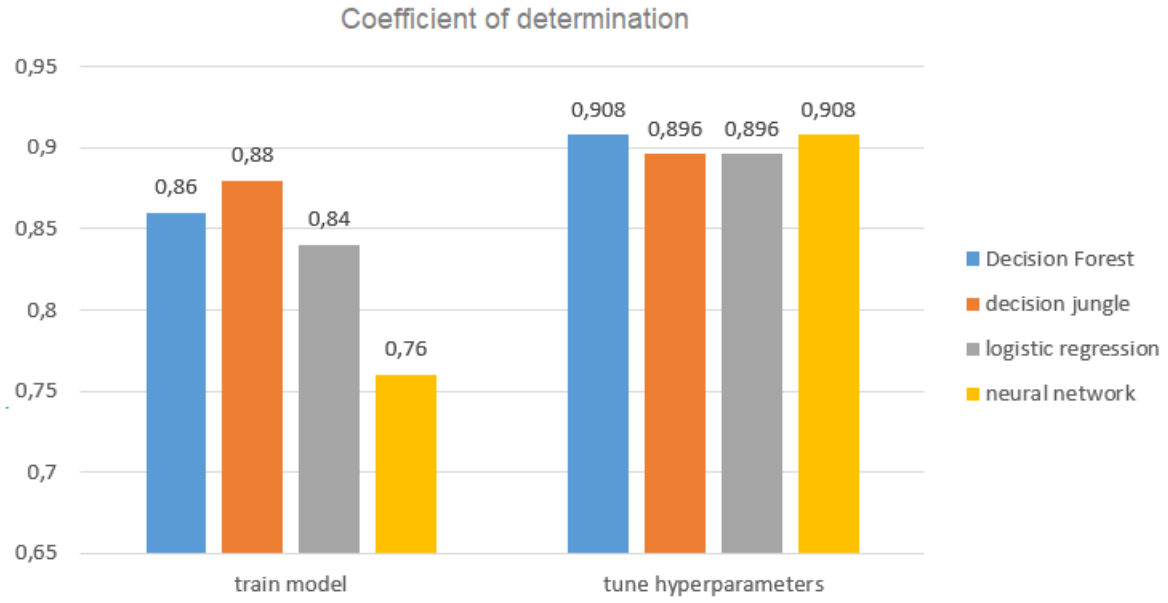
In Figure 9.32 it is possible to observe the metrics and the confusion matrix of the set formed by train model, while in Figure 9.33 the results come from the cross-validate type. In this algorithm it is clear that the model with *tune hyperparameters* is much better than the model with only the train model, because, while the accuracy in the simple one is 76%, the accuracy in the complex one is more than 90%. According to the results obtained with the train model set, it is possible to mention that this type of algorithm cannot be used because the percentage is too low and moreover, it is not balanced. In fact, it could be good because the model is successful when it has to decide neither the tool has to be replaced or not, but in the other labels it makes a lot of mistakes and this is not acceptable. The differences between the scored labels and the real values are in Appendix 5.

9.6.5. Comparison between algorithms

Once all the results have been explored, the first clear decision is about the training and testing sets. One section before has been mentioned that it was possible to divide the experiment with the same algorithm in two parts. One part was training the data with a simple configuration and, the other one trained it with more complexity. Regarding the results achieved, it is obvious to detect that it is better to treat the data more complexly, because the results are always higher than the other way of studying. Thus, the set formed by *split data- train model- score results* could be eliminated of the model because it is less accurate and the results are not as expected, as it can be seen in Graph 9.1. Finally, the selected training mode is the one formed by *partition and sample-tune hyperparameters-cross validate model,* since it could afford the best results making lots of combinations. It is important to note that the accuracy with this configuration is never less than 88% and it means that the model is working very well.

On the other hand, it is difficult to decide which algorithm is the best to apply, using the classification task, since all of them have more than 89% of overall accuracy and it means the machine learning process is going ahead (Graph 9.1). However, the most interesting labels in classification are *CHANGE and TAKE CARE*, because these two labels are saying that the tool is wearing out and probably it would need a replacement soon. Observing these labels, the less accurate algorithm is logistic regression. This algorithm is less precise than the others, because the two important labels are lower than 90%, while the label *NO PROBLEM* is higher and it contributes to raise the average of all the model. For this reason, logistic regression would be the worst model and it would not be an option in the classification task.

Finally, among all the other algorithms, it is difficult to choose only one because all of them exceed 90% in the most conflictive label (change) and, it would be possible to work with any of them. Probably, the decision of choosing only one could be more precise if there was more data to work on.



Graph 9.1: Comparison of all the coefficients of determination

9.7. Regression

The second type of algorithms is described by the regression. In general, a regression algorithm tries to learn the value of a function for a particular instance of data. There are a lot of methods to apply it, but after doing researches on the best algorithms, the conclusion is that there are four that could be trained and improve the model to get the correct results. As in the classification models, these algorithms have also been done with both types of configurations to observe which method of training is the best (train model or tune hyperparameters).

On the other hand, in Azure's studio when the regression algorithms are applied, the way to realize if the model is correct or not is not the same as in the classification. While in the classification the main goal is the accuracy, in the regression there are other items. Mean absolute error, root mean squared error, relative error and the coefficient of determination are the values observed to determine if it is possible to work with this model.

Finally, this section will be separated differently than classification, because there was no way to program different regressions for different materials. Therefore, there will be three types of regression. The first one contains all that data without separating them by materials and then, there will be one regression for each material (1 and 2).

The four algorithms selected to carry out the regression are Boosted Decision Tree Regression,

Decision Forest Regression, Neural Network Regression and Linear regression. Appendix 6 shows the structure of the model in Azure's platform.

9.7.1. Regression with both materials

9.7.1.1. Boosted Decision Tree Regression

Boosting means that each tree is dependent on prior trees. The algorithm learns by fitting the residual of the trees that preceded it. Thus, boosting in a decision tree ensemble tends to improve accuracy with small risk of less coverage. Moreover, this regression method is a supervised learning method, and therefore requires a labelled dataset.

On the other hand, boosted decision trees use an efficient implementation of the MART gradient boosting algorithm. It builds each regression tree in a step-wise fashion, using a predefined loss function to measure the error in each step and correct it for the next one. Thus the prediction model is actually an ensemble of weaker prediction models.

As classification methods, Boosted Decision Tree regression has been done with both types of training to know which one is the best.

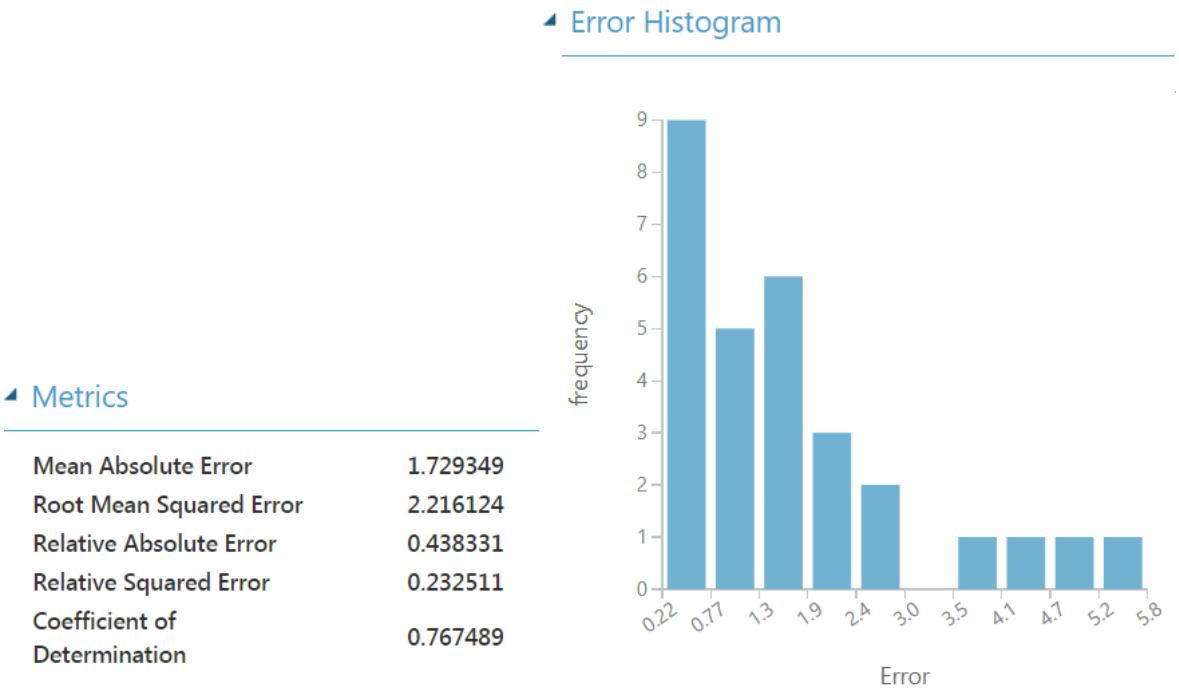


Figure 9.34: Results obtained with train model

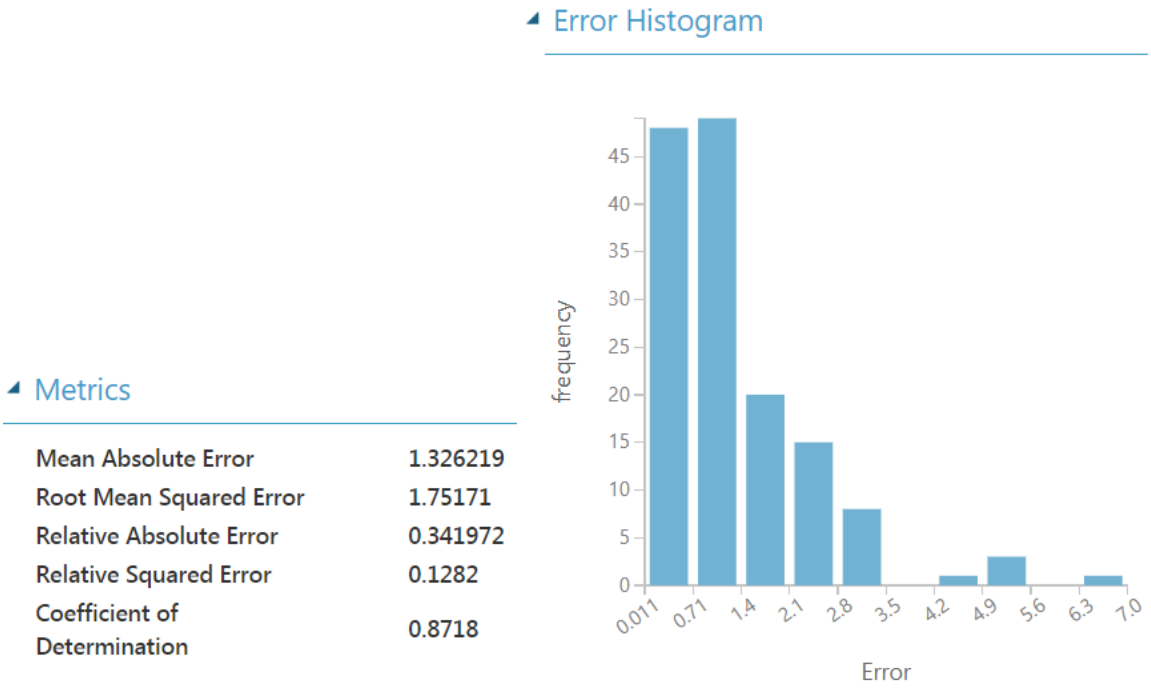


Figure 9.35: Results obtained with tune hyperparameters

In Figure 9.34 it is possible to observe the results extracted from the train model module and Figure 9.35 shows the results obtained with tune hyperparameters. Observing both results is obvious that the most complex one (tune hyperparameters) is better because the coefficient of determination and the mean absolute error are respectively higher and lower. Even though the coefficient does not overcome 90%, the model is working correctly and, at the moment, this would be a good model to apply. However, the model obtained by simple train does not show good statistics because it is not capable to exceed 77%.

9.7.1.2. Decision forest regression

This type of algorithm are non-parametric. They perform a sequence of simple tests for each instance, until it reaches a leaf node that means getting a decision. In addition, it is very efficient in computation and memory using, while it is training and predicting. Also, decision trees can represent non-linear decision boundaries and at last, it performs integrated feature selection and classification and it is resilient against the noisy features. Regarding how it works, this regression consists of an ensemble of decision trees, where each of them outputs a Gaussian distribution as a prediction.

The results of both training sets (train model and tune hyperparameters) are shown below.

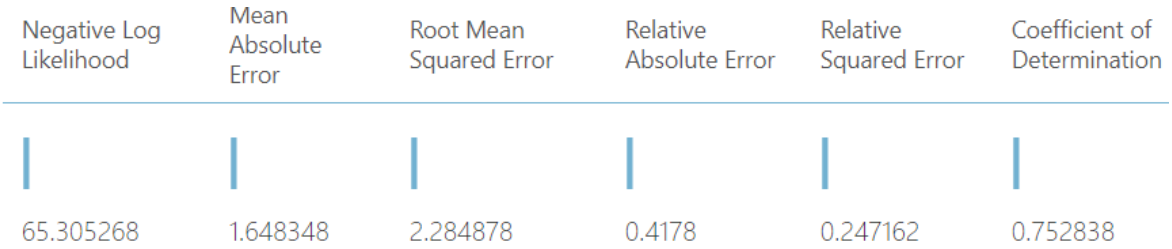


Figure 9.36: Results obtained with train model

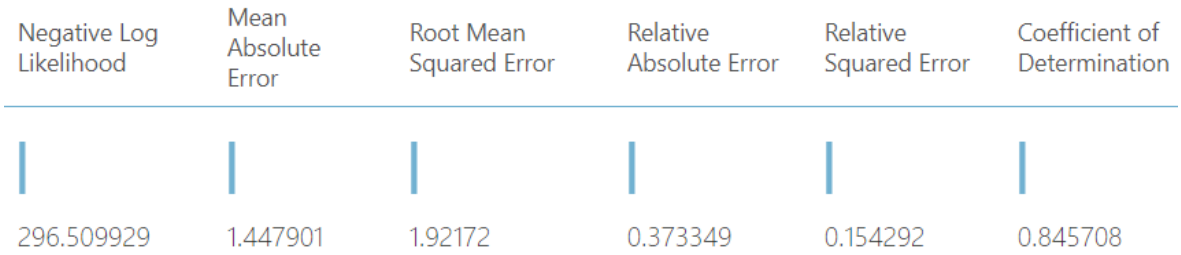


Figure 9.37: Results obtained with tune hyperparameters

Figure 9.36 shows the results from the simple train model, while Figure 9.37 shows the result from tune hyperparameters. Observing the error and coefficient of determination, it is easy to affirm that the second one is the best, because its parameters are better. Even though the coefficient of determination is high, it is lower than the best one from Boosted. Indeed, it could be taken into account because with more data, the results would change and it would not be eliminated from the studies.

9.7.1.3. Linear regression

The aim of linear regression is to establish a linear relationship between the variables that could explain the behaviour of the model. Moreover, it is a common statistical method that has been introduced in machine learning for fitting the line and measuring the error [19]. The main objective is to find and predict a numeric target and it works very well on high-dimensional, sparse datasets lacking complexity.

This module is able to solve some types of regression. The first one would be simple regression and it consists to look for the relationship between an independent variable and a dependent one. The second one could be the multiple linear regression, that has the same meaning of simple regression but, instead of comparing one variable of each kind, it involves two or more independent variables with only one dependent.

Measuring the error well is crucial to achieve good results and there are two ways to calculate the error. The first one is the Gradient descent and it minimizes the amount of error at each step of the model training process. Otherwise, there is Ordinary least squares that refers to the loss function and it computes error as the sum of the square of the distance from the actual value to the predicted line and fits the model by minimizing the squared error.

Once the meaning of Linear Regression is clear, the results from the experiment are shown to extract some conclusions.

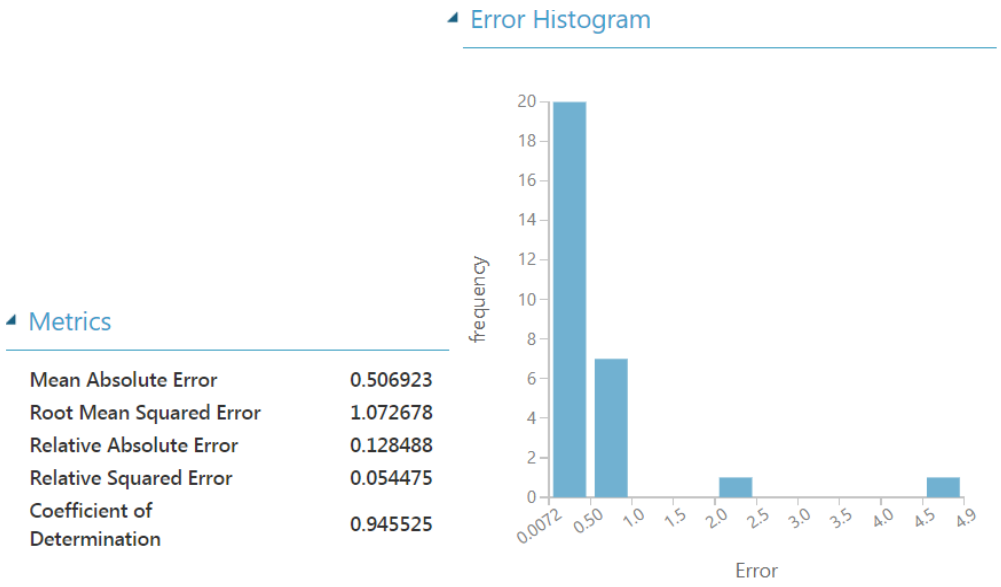


Figure 9.38: Results obtained with train model

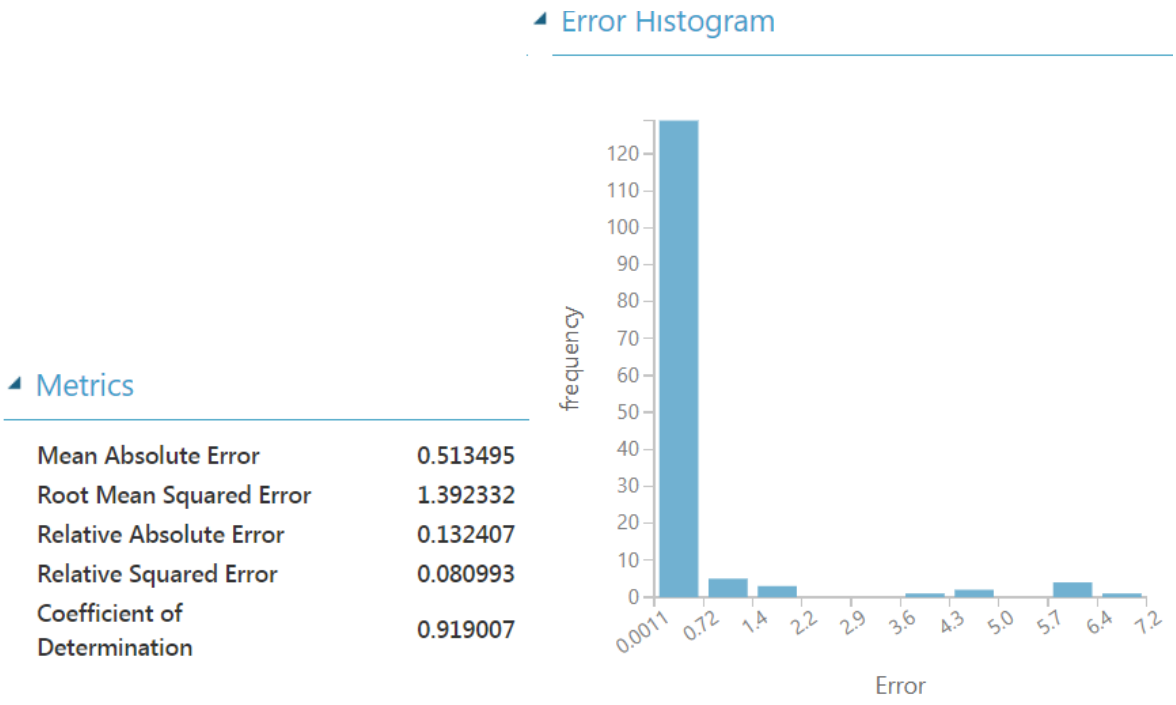


Figure 9.39: Results obtained with tune hyperparameters

Observing Figure 9.38 and Figure 9.39 it is possible to realize that something strange has happened comparing with all the other algorithms. While *tune parameters* is the best way of training in all the other methods, this is not the case here. One explanation is that the set configured with *train model* and the other components has better parameters than the other one. Even though the difference between two models is not very relevant, the train model configuration is better. The reason for this strange behaviour could be explained by the size of the dataset. More data would be needed to approach better results. Finally, both kinds of training have to be considered because the coefficient of determination and the mean absolute error are better than the other regression's methods. Both coefficients of determinations are higher than 91% while the error is less than 0.52 in each case.

9.7.1.4. Neural network regression

Neural network regression is one of the most useful algorithms in terms of deep learning and modelling complex problems such as image recognition, as it was mentioned in the classification section. Moreover, this method is easily adapted to regression problems and sometimes it is the best way to get the optimal results. Adaptive weights and approximation of non-linear functions of their inputs are the requirements for applying this algorithm. Therefore, neural networks are able to solve problems that could not be solved by simple regressions.

Next figures show the results of both types of training.

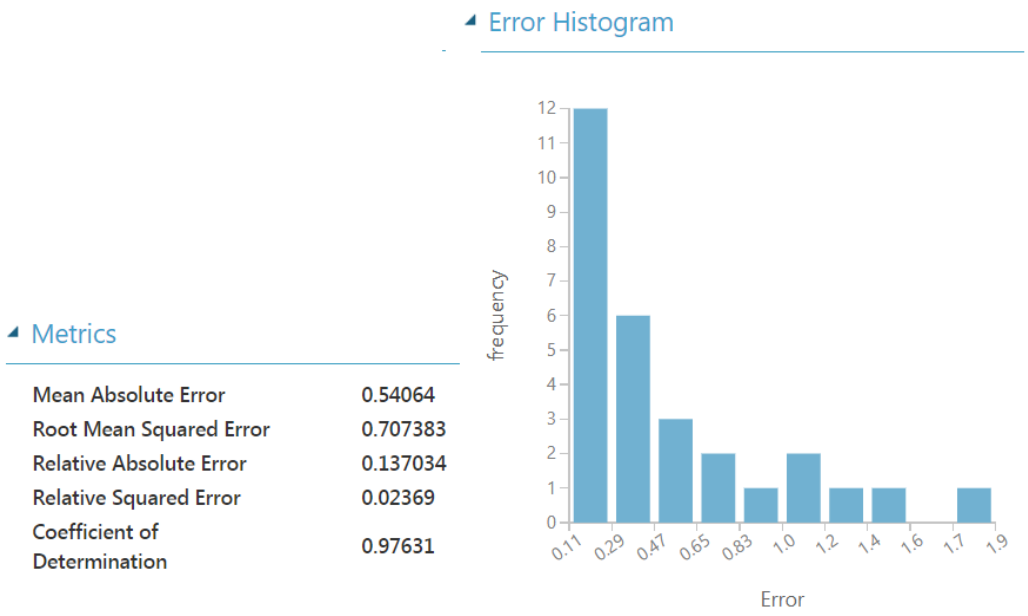


Figure 9.40: Results obtained with train model

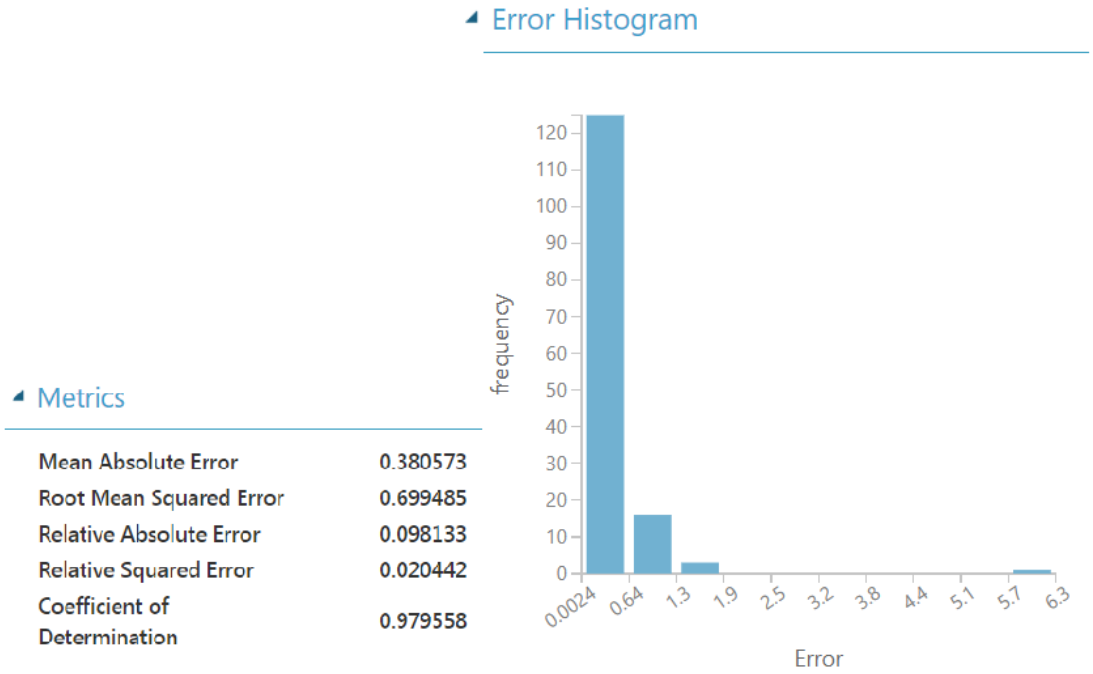


Figure 9.41: Results obtained with tune hyperparameters

and Figure 9.41 it is obvious that neural networks algorithm is the best for regression because its error and the coefficient of determination are better than the others. In terms of choosing the best training method, both cases would be accepted since the error and the coefficient are respectively low and high. Getting a coefficient of 98% means that the model is very well implemented and it could be used to predict the tool wear almost perfectly. The differences between the scored labels and the real values are in Appendix 7.

9.7.2. Regression with material 1

Once the regression with both materials has been analysed, it would be well considered to carry out studies for each material. Doing it separately could contribute to detect which material has more variance. In fact, the model will only be trained by *tune hyperparameters*, since it is the one that offers more accuracy, precision and better results in terms of error and coefficient of distribution in almost all cases. Below are exposed all the results. After analysing them, they will be discussed and compared with the results obtained from the regression with both materials.

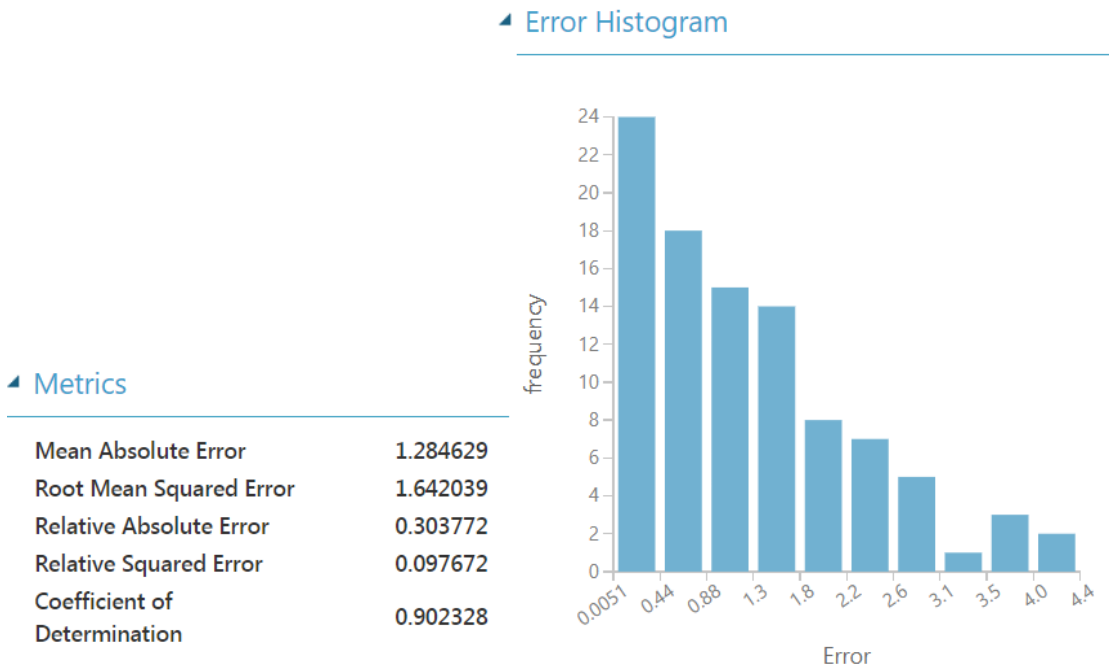


Figure 9.42: Results obtained with tune hyperparameters from Boosted decision tree

Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
207.391253	1.562074	2.01186	0.369378	0.146621	0.853379

Figure 9.43: Results obtained with tune hyperparameters decision forest

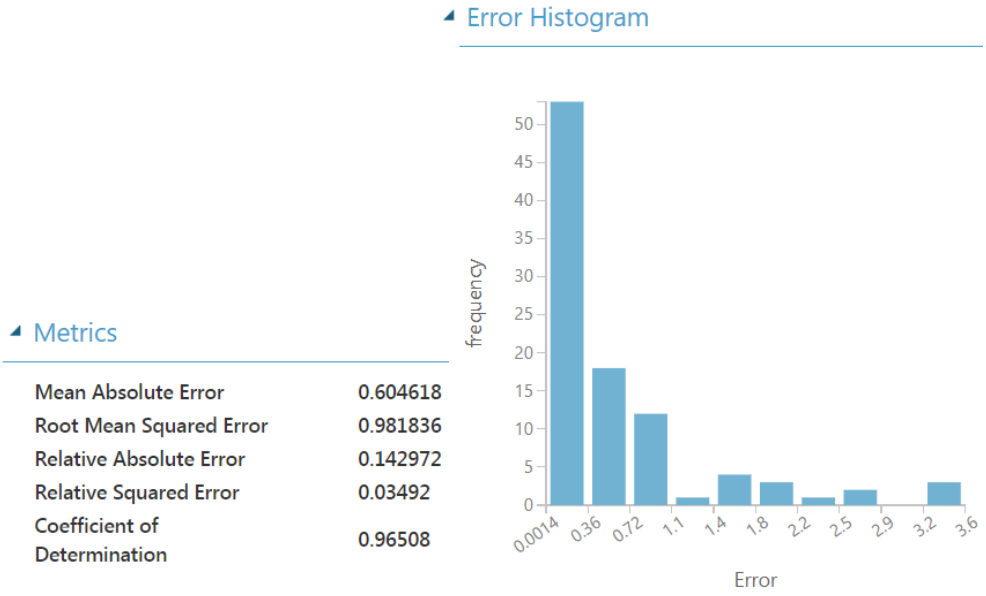


Figure 9.44: Results obtained with tune hyperparameters from linear regression

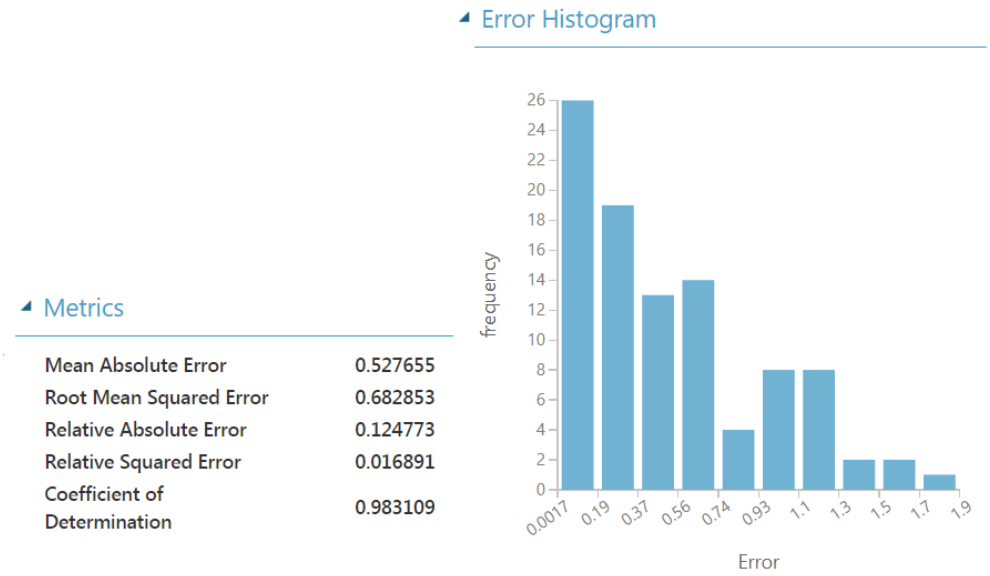


Figure 9.45: Results obtained with tune hyperparameters from Neural Network

Observing, analysing and comparing these results shown in Figure 9.42, Figure 9.43, Figure 9.44 and Figure 9.45, with those obtained from the regression with all the data, it is possible to ensure that for material 1, it would be more useful to perform the regression separately. Regarding the results obtained, all of them overcome the results from the regression with both materials and moreover, their percentages of the coefficient of determination and the mean absolute error are respectively higher and lower. Even though the only one that has a percentage lower than 90% is Decision Forest Regression, the others have accuracies higher than 90% and all of them could be applied to work

with machine learning but, as in the regression with both materials, the best algorithm is Neural Network regression since it has the lowest absolute error and its percentage is higher than 98%.

9.7.3. Regression with material 2

In this section, the same results as in section 9.6.2 will be extracted to analyse the differences between the results from the other two regressions. Carrying out this study can provide more knowledge about the features to take into account. Detecting if the material is one of the inputs that can change the statistics is important as well and this study will focus on this. These regressions will be trained by *tune hyperparameters*, because it is the training that achieves better results. Next figures show the results and they will be compared with the others extracted from the other regressions, once these results will be analysed.

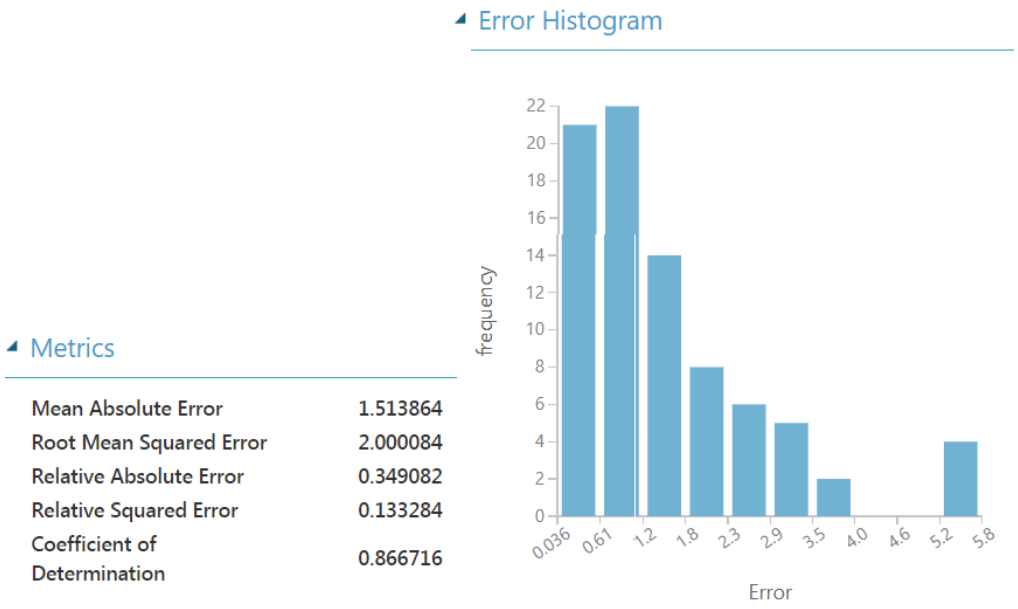


Figure 9.46: Results obtained with tune hyperparameters from Boosted decision tree

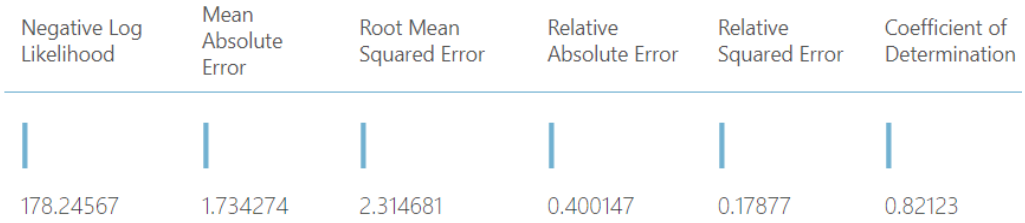


Figure 9.47: Results obtained with tune hyperparameters from Decision forest

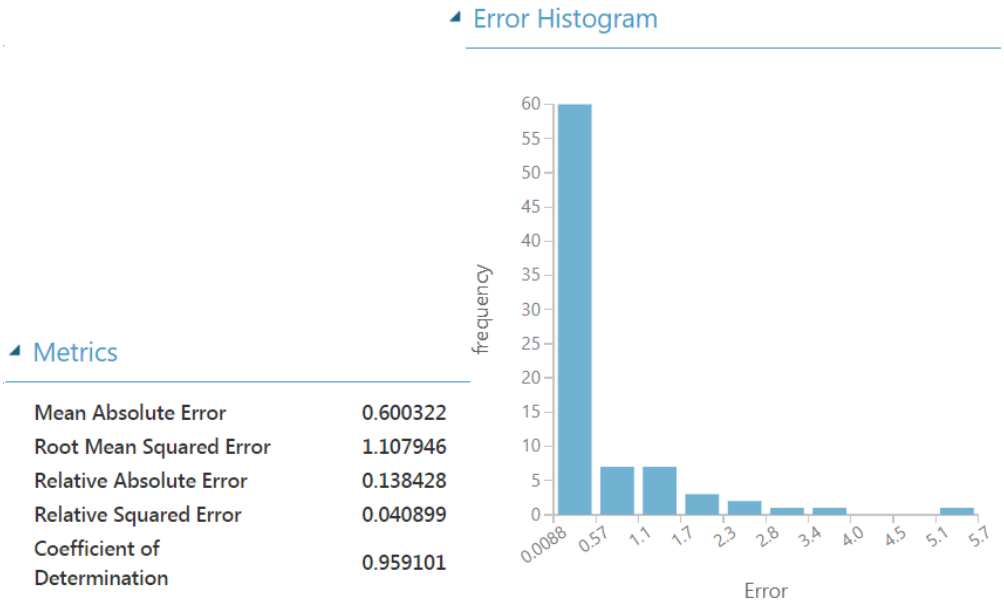


Figure 9.48: Results obtained with tune hyperparameters from linear regression

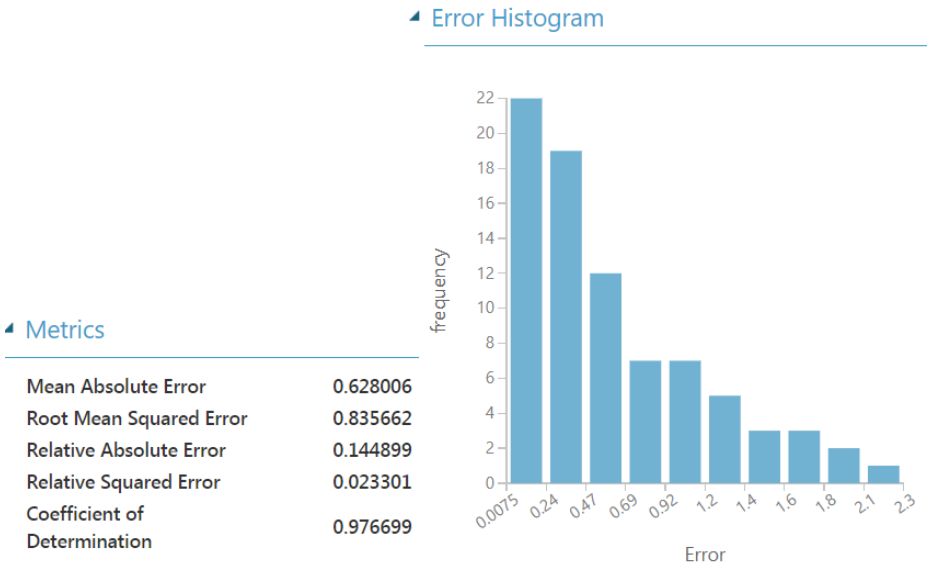


Figure 9.49: Results obtained with tune hyperparameters from Neural Networks

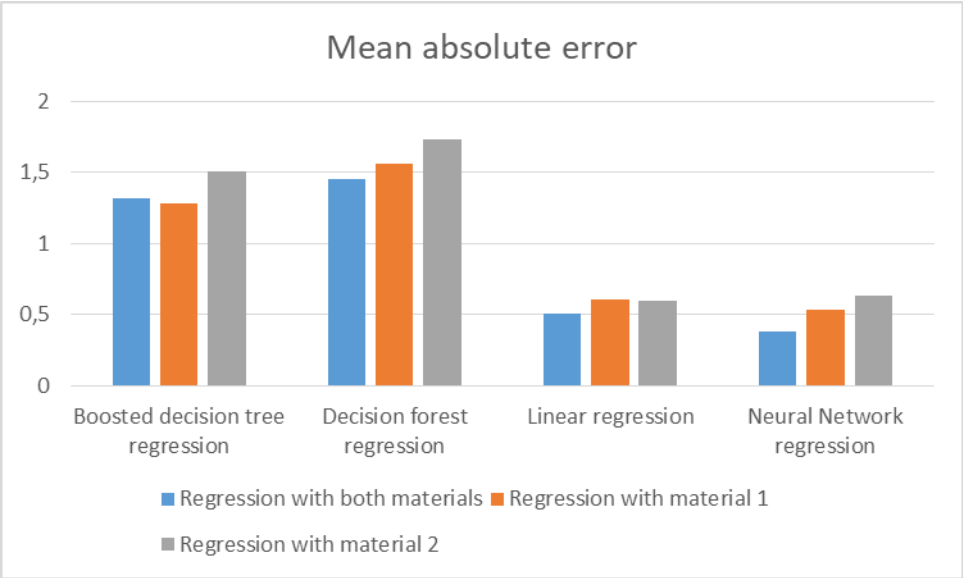
Comparing these results shown in Figure 9.46, Figure 9.47, Figure 9.48 and Figure 9.49 with those obtained from the regression with both materials and regression with material 1, it is possible to detect that these results are similar to the regression with all data. They are a little bit lower than the regression with material 1. These differences could appear because material 2 has more variance than material 1, thus the precision would not be the same, or it could also be generated because there are less data than material 1 and the accuracy is not the same. Even though the results are a little bit lower for regression with material 2, these are very good and Neural Network regression could be applied to get the optimal results.

9.7.4. Comparison between all the solutions

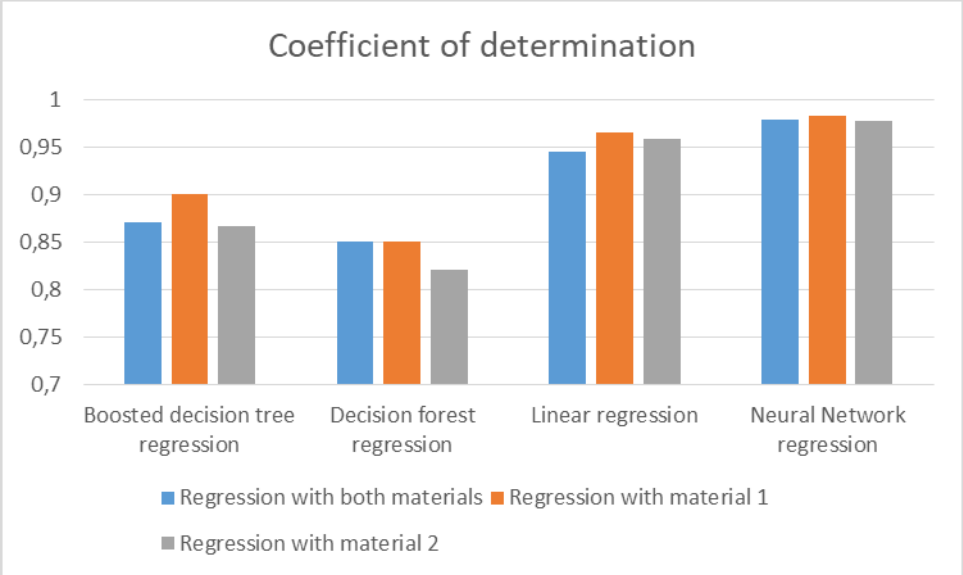
In section 11.5, it was not obvious to define which algorithm was the best to apply because three of them had good precision and accuracy and it could find the optimal results in a successful way. Nevertheless, this problem does not appear in regression because there are two algorithms that are better than the others. In fact, one of them is the best and this will be the chosen one for being applied and solving the problem through machine learning.

Regarding the results shown in Graph 9.2, Graph 9.3 and the other statistics obtained before it is clear that the best regression algorithms are Linear regression and Neural Network regression. These two methods are the best because the mean absolute error is lower in all cases and, in addition, the coefficient of determination is higher in all cases. For this reason, there is no doubt that one of them should be applied to get the best results. Indeed, the algorithm selected will be Neural Network since it is better than Linear regression. It always achieves more than 97% of efficiency and this the main argument to select Neural Networks.

On the other hand, there are other statistics that should be considered. In regression model, three different types of regression, depending on the material selected, has been done. All the statistics are similar between these three different methods, but observing the error and the coefficient of determination of the regression obtained from material 1, it could be a better option to study each material through different dataset to improve the results. Even though the regression with material 2 does not improve the results from the regression with both materials, this phenomenon could be assigned to the data. Carrying out this study with more data could improve the results because the errors would not be as high as in this case.



Graph 9.2: Comparison of the errors obtained from all the regressions



Graph 9.3: Comparison of the coefficients of determination fr+om all the regressions

Conclusions

During the performing of this thesis, a lot of knowledge has been acquired because some of the topics that have been involved were completely or at least partly unknown.

First of all, an interesting topic analysed in this project is the data mining and all the information that it could provide if the data are treated well. Knowing all the hidden patterns of the data would allow to have a better understanding of the mistakes produced in the real world and it would help to improve the efficiencies of all kind of processes. Moreover, the data can be treated in many ways, but classification, association, clustering and regression are the most used among the others. Classifying the data into some different labels and predicting the exact values are some of the processes that can be done by joining data mining with machine learning.

Moreover, the most important concept to keep in mind is machine leaning because without it, nothing in this project would make sense. Machine learning is the future since with it, it would be possible to predict some values and to know some tendencies, but this is not an easy work. First of all, the problem has to be understood and the data should be treated perfectly to find the correct patterns. Once it is achieved, there are a lot of algorithms that could be used but not all of them could be applied. Depending on the data that are used, some algorithms will be more appropriate than others. If the algorithm is applied well and the model finishes satisfactorily, then companies could benefit from this opportunity.

With these advantages a lot of companies would take advantage of machine learning because they would be able to know what happened when a problem appeared, they would know which tendencies are better and what they should do to increase its benefits. In one word, it would mean more knowledge for the companies about themselves. Introducing machine learning in the companies is difficult because it requires Smart factories and it entails some changes in the intern and extern infrastructure of the companies.

Smart factories tries to build an automatized and collaborative world since their aim is to combine the skills of the robots and the skills of the humans. In fact, these factories are the connection between the two worlds, the physic world and the virtual world and increasingly, machine learning is being applied in them. There are a lot of problems in an industry, such as the planning production, the control of the warehouse and the failure analysis, among others. Focusing in failure analysis of the tool, the tool wear is the most critical parameter and it could depend on several features.

According to the problems mentioned about the companies, one of them is the topic of this thesis.

Smart factories and failure analysis are the topics of this thesis and the main goal of this project was to find some patterns that could explain the wear of the tool and which parameters had to be considered to make a good analysis.

First of all, the data has to be understood to know which variables appear. It is important to detect which variable will be selected to validate the model because the other ones will be trained to achieve the expected results. In addition, sometimes the variable to validate the model is not in the dataset and it has to be created depending on the requirements.

In this case, the dataset given was not in the best conditions. There were some outliers that had been removed, the data was transformed and some variables have been created to get the best results. According to the papers read, the best algorithms to apply for this case (failure analysis) were classification and regression. For this reason, the project has been divided into two types of experiments.

The first experiment was with classification algorithms. There were four algorithms available according to the data used and two different kinds of trainings depending on the complexity needed. In fact, as each algorithm had to be modelled with both training methods, at last there were 8 different experiments to be compared between them. The predictable attribute in these cases were the operating state of the tool. There were three different labels depending on the wear of the tool. These three different classes were *No problem, take care* and *change* and if the predictable value was change, it means that the tool had to be replaced because it could not work correctly. Once the four algorithms were applied, the statistics showed that three of the four algorithms used were good to achieve the optimal results. Decision Forest, Decision Jungle and Neural networks were algorithms that could be applied to this problem and all of them solved it with 90% of accuracy. Obviously, the best training model was tune hyperparameters because it explored more nodes and finally it found the best results.

On the other hand, the second types of experiments were done by algorithms of regression. Four algorithms were considered and they followed the same structure in terms of training but the data were divided into different sets because exploring different materials together or not could affect the final decision. Thus, instead of doing only 8 experiments, there were 16 experiments. The first experiment was done with all the data, including both materials. The results showed that the best algorithm was Neural Networks because its error was the lowest and moreover its coefficient of determination was the best one. It reaches 98% of efficiency.

Comparing these results with those obtained from the regression with material 1 and regression with material 2, it is possible to detect that the results are very similar. While in regression with

material 1, the best algorithm is also Neural Network and the coefficient of determination is of 0.3% higher than the first one, in regression 2 this coefficient is a little bit lower than the one obtained with both materials. From these observations, two different ways could be taken. The first one is that the material does not influence the final results because the parameters are almost the same. However, the second observation mentions the possibility that the material could influence the final decision because the regression with material 1 is better in terms of coefficient of determination but worse in terms of error. One of the reasons that could explained this phenomenon is that the dataset for each material is too small. Few values are plotted and then, the error affects the results more negatively. For this reason, it seems that with more data, the results for different dataset could be improved.

Finally, the best algorithm regarding the results obtained from all the experiments would be Neural Networks, because it is useful in both cases, classification and regression.

Acknowledgments

First of all, I would like to thank the professor Giulia Bruno because she was the person who solved all the problems that I have, she gave me some advice about how to approach the thesis and she followed up our project. Without her help, my task would have been more difficult and I would have spent more time to solve my troubles.

Moreover, I would also like to thank the professor Franco Lombardi because he gave us the opportunity to work with him and to develop a topic that I was interested in. Since we arrived in Torino, we had some problems to find a thesis until we went to Mister Lombardi's office. He was very kind because he met us in his office and he lost his time listening our project's ideas. Finally, he recommended some other topics that could be more interesting for both parts.

Finally, my partner Andrés Lizán supported me when I had some problems. He played an important role in the experimental part since we worked with the same programs.

Thank you

Sergi Fajas

References

[1] MEHMED KANTARDZIC, *Data Mining: concepts, models, methods and algorithms*. New York (Chichester) 2002.

[2] INTENSIVE WORKING GROUP OF ACM SIGKDD CURRICULUM COMMITTEE. *Data Mining Curriculum: A Proposal* [http://www.kdd.org/curriculum/index.html, June 11, 2018].

[3] FAYYAD, USAMA, G. PIATETSKY-SHAPIRO, AND PADHRAIC SMYTH. *From Data Mining to Knowledge Discovery in Databases.* AI Magazine, 1996, p 37–54.

[4] PANG-NING TAN, MICHAEL STEINBACH, ANUJ KARPATNE, VIPINKUMAR, *Introduction to Data Mining. Chapter 3: "Classification: Basic Concepts and Techniques."* 2018, p 113–92.

[5] DORMEHL, LUKE. *What Is an Artificial Neural Network? Here's Everything You Need to Know.*"Digital Trends. [https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/), June 1, 2018].

[6] X. WU, G. GUO, AND J. WENG, *Skull-closed Autonomous Development: WWN-7 Dealing with Scales*, Proc. International Conference on Brain-Mind, July 27–28, East Lansing, Michigan, pp. 1–9, 2013.

[7] TCHIRCOFF, ANDREW. *The Mostly Complete Chart of Neural Networks, Explained.* [https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464), June 1, 2018]

[8] TREVOR HASTIE, ROBERT TIBSHIRANI, JEROME FRIEDMAN. *The elements of Statistical Learning*. Springer Series in Statistics, Pg. 134. Stanford, California, August 2008.

[9] THEODORIDIS, SERGIOS*. Pattern Recognition*. Elsevier B.V. p. 203, ISBN 9780080949123, 2008.

[10] ASA BEN-HUR, DAVID HORN, HAVA SIEGELMANN, AND VLADIMIR N. VAPNIK. *Support vector clustering.* Journal of Machine Learning Research, p. 125–137, 2001.

[11] PANG-NING TAN, MICHAEL STEINBACH, ANUJ KARPATNE, VIPINKUMAR, *Introduction to Data Mining. Chapter 5: "Association Analysis: Basic Concepts and Algorithms."*, 2018, p 357–449

[12] PANG-NING TAN, MICHAEL STEINBACH, ANUJ KARPATNE, VIPINKUMAR, *Introduction to Data Mining. Chapter 7: "Cluster Analysis: Basic Concepts and Algorithms."*, 2018, p 525-612

[13] DIEDERIK VERZIJL, KRISTINA DERVOJEDA & FABIAN NAGTEGAAL, PWC NETHERLANDS, AND LAURENT PROBST & LAURENT FRIDERES, PWC LUXEMBOURG, *Smart process applications*. Union European: Business Innovation Observatory, 2014

[14] DGH (líder), LEYTEC (sensores), VIRTUALWARE (simulación virtual), IBERMATICA (TIC, logística, navegación), FICOMIRRORS (escenario logística inteligente), CESA (escenario manipulación colaborativa) y CONTINENTAL (escenario robótica colaborativa), *Smart Factory, sistemas inteligentes para las fábricas del futuro,* Madrid 2016. [http://virtualwaregroup.com/es/noticias/smart-factory-sistemas-inteligentes-fabricas-del-futuro, April 28, 2018]

[15] JINJIANG WANGA, YULIN MAA, LAIBIN ZHANGA, ROBERT X. GAO B, DAZHONG WUC, *Deep learning for smart manufacturing: Methods and applications*, China & USA: 2018

[16] R.PORTUGAL ("Industrial Engineering teacher"), *Desgaste y vida de las herramientas de corte*, Perú, 2000

[17] JOSÉ ANGEL MARAÑÓN, *Los errores del mecanizado*, Barcelona 2014 [http://www.interempresas.net/MetalMecanica/Articulos/116756-Los-errores-en-el-mecanizado.html, May 4, 2018]

[18] DAZHONG WU, CONNOR JENNINGS, JANIS TERPENNY, ROBERT X. GAO, SOUNDAR KUMARA, *A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests.* Vol 139, USA: 2017

[19] GARY ERICSON, *What Is Azure Machine Learning Studio?, USA 2015* [ttps://docs.microsoft.com/en-us/azure/machine-learning/studio/what-is-ml-studio%0Ahttp://azure.microsoft.com/en-us/documentation/articles/machine-learning-what-is-ml-studio, June 2018] .

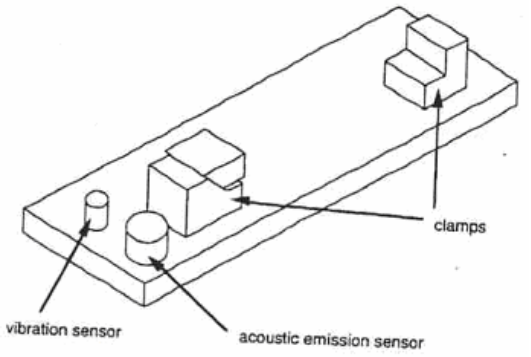
Appendixes

Appendix 1: New opportunities of digitalization

Digitalization is already changing every aspect of life and existing business models. It enables industry to turn its product ideas into reality in a new way by drawing on technological trends like generative design and intelligent models. Additive manufacturing and advanced robotics put production on an innovative footing, and the use of cloud solutions and knowledge automation lets machine builders develop new service models. Some of the aspects that could be improved will be the following.

Appendix 2: Detailed setup description

Mounted to the table is an acoustic emission sensor model WD 925 (PHYSICAL ACOUSTIC GROUP,frequency range up to 2MHz). The acoustic emission sensor is glued to a custom made base which in turn is attached to the clamping support. The layout of the sensors on the clamping device is shown in the next figure



The signal from the acoustic emission sensor goes into the single ended terminal of an acoustic emission preamplifier (DUNEGAN/ENDEVCO, model 1801 with integrated 50KHZ high pass filter). The signal is then amplified by a dual amplifier DE model 302A (DUNEGAN/ENDEVCO). The signal is then fed into the RMS meter which is a custom made device built by the LMA of the University of California at Berkeley. The time constant is set to 8.0ms. The signal is then fed into the PHOENIX CONTACT UMK-SE 11,25 cable which feeds the signal into a MIO-16 high speed data acquisition board (National Instruments). The data acquisition board is mounted in a IBM PC 486DX/2-66.

Also mounted on the clamping device on the table is a vibration sensor, an accelerometer (model 7201-50, ENDEVCO) with a frequency range up to 13KHz. Its signal is fed into an ENDEVCO 104 charge amplifier with sensitivity 5.71 and 100mV/g output. The signal is then fed into an ITHACO 4302 DUAL 24dB/octave filter with corner frequencies 400 Hz and 1KHz. Following is a RMS meter same make and settings as described earlier. The signal is then fed then through the PHOENIX CONTACT UMK-SE 11,25 cable connector into the high speed data acquisition board of the computer.

The same vibration sensor that is mounted on the table is also attached to the spindle into a pre-existing threaded hole close to the tool. The signal follows the same path as described for the other vibration sensor except that the signal is fed into the PHOENIX CONTACT cable connector.

Signals from another acoustic emission sensor mounted into another threaded hole on the spindle next to the tool is fed into the differential terminal of an acoustic emission preamplifier model 1801 (DUNEGAN/ENDEVCO) and then follows the same path as outlined for the other acoustic emission sensor.

A OMRON K3TB-A1015 current converter, powered by a HP 6237B triple output power supply providing 15V, feeds the signal from one spindle motor current phase into the cable connector.

A model CTA 213 current sensor (Flexcore Div. of Marlan & Associates, Inc.) which uses the same

phase of the spindle motor current is fed into the cable connector.

Terminals 33 (ground) and 38 (+5V) are used with a switch to trigger the data acquisition.

With industrial applicability in mind, a 70mm face mill with 6 inserts (Figure 3) was chosen as the tool. The inserts KC710 was selected based on the recommendations for roughing (Kennametal, 1985). KC710 is coated with multiple layers of titanium carbide, titanium carbonitride, and titanium nitride. (TiC/TiC-N/TiN) in sequence. These layers retain the toughness of tungsten carbide but have improved resistance to cratering and edge wear. At the same time, they have the advantage of titanium carbide plus reduced face friction. This insert is recommended for heavy roughing.

Appendix 3: Discrete Fourier Transform

These are the transform for continuous-time, infiniteduration signals (FT), and for discrete-time, finite-duration signals (DFT). The FT is most amenable to analytical insights and manipulations, and is used in analyzing analog systems. The DFT the workhorse that is widely used in digital computations, because of a fast algorithm called the FFT (fast Fourier transform) for computing the DFT. For convenience, we repeat the definitions of the forward and inverse transforms here. For the FT, both the time variable t and frequency variable ω are continuous-valued and vary from −∞ to ∞. The transform equations are

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt \qquad \text{forward transform}$$

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\,d\omega \qquad \text{inverse transform}$$

The DFT is used in the case of a finite-duration, discrete-time signal, x[0], x[1], . . . , x[N− 1]. The DFT X[0], X[1], . . . , X[N − 1] is also finite-duration and discretefrequency. The definitions of the forward and inverse DFT are:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-jk\omega_0 n} \quad k = 0, 1, \ldots, N-1$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jk\omega_0 n} \quad n = 0, 1, \ldots, N-1$$

$$\omega_0 = \frac{2\pi}{N}$$

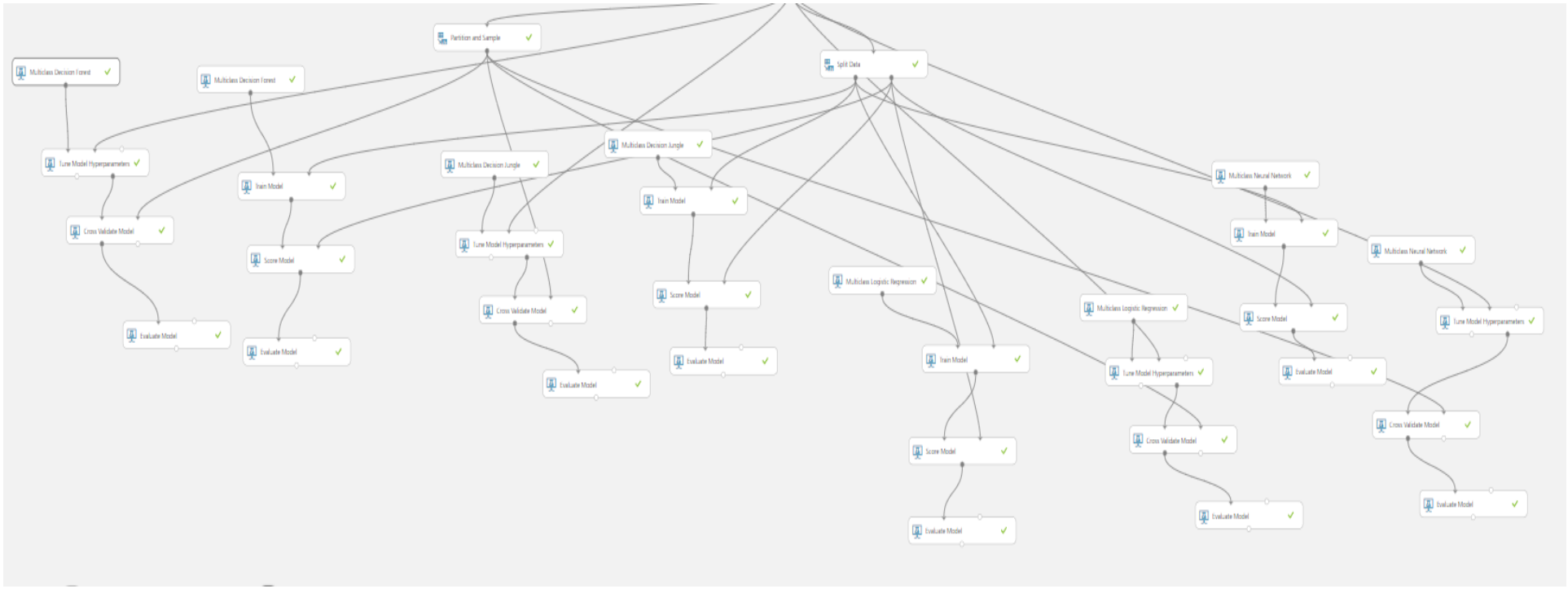
Appendix 5: Differences between the real values and scored values of Neural Networks algorithm in classification.

nivells	Scored Probabilities for Class "change"	Scored Probabilities for Class "NO problem"	Scored Probabilities for Class "take care"	Scored Labels
NO problem	0.000001	0.025122	0.978713	take care
take care	0.007217	0.026555	0.624462	take care
take care	0.000083	0.000611	0.994699	take care
change	0.974749	0.006706	0.0001	change
change	0.999907	0.000087	0.000048	change
change	0.999968	0.000002	0.00039	change
change	0.999441	0.000001	0.012169	change
NO problem	0	1	0	NO problem
NO problem	0.000006	1	0	NO problem
NO problem	0	1	0.000031	NO problem
NO problem	0	1	0.000092	NO problem
take care	0.000047	0.295945	0.526824	take care
take care	0.065788	0.000099	0.96583	take care
take care	0.00456	0.000001	0.999374	take care
change	0.99997	0	0.000551	change
change	0.999995	0	0.000327	change
NO problem	0	1	0	NO problem
NO problem	0	1	0	NO problem

Appendix 7: Differences between the real values and scored values of Neural Networks algorithm in regression.

life	Scored Labels
16	14.882794
13	12.876986
11	10.927834
10	9.904192
9	8.944924
8	7.394121
7	7.23098
6	6.29959
5	4.91279
4	4.049487
3	2.769835
9	9.156713
7	7.187311
6	6.249987
5	4.469437
4	4.04724
3	3.123272
2	1.972136
1	1.010448
0	0.098347
15	13.551874
14	13.863857
13	12.533648

Appendix 4: Structure of classification algorithms



Appendix 6: Structure of regression algorithms